Universidade Federal do Rio de Janeiro

Núcleo de Computação Eletrônica

Edinilson Machado Coelho

REDES P2P:

Analise de aplicativos P2P

Rio de Janeiro

2008

Edinilson Machado Coelho

REDES P2P:

Analise de Aplicativos P2P

Monografia apresentada para obtenção do título de Especialista em Gerência de Redes de Computadores no Curso de Pós-Graduação Lato Sensu em Gerência de Redes de Computadores e Tecnologia Internet do Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro – NCE/UFRJ.

Orientador:

Prof. Moacyr H. Cruz de Azevedo, M.Sc., UFRJ, Brasil

Rio de Janeiro

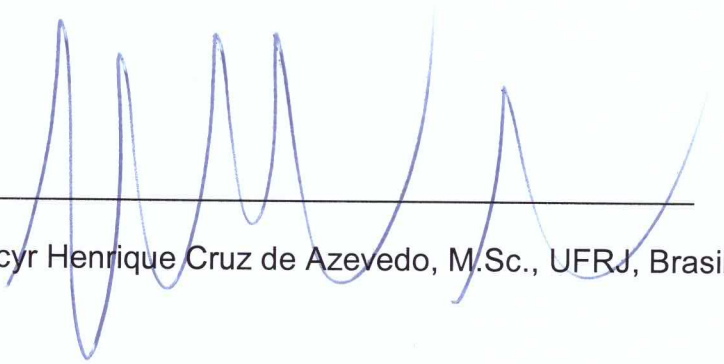
2008

Edinilson Machado Coelho

REDES P2P : Analise de aplicativos P2P

Monografia apresentada para obtenção do título de Especialista em Gerência de Redes de Computadores no Curso de Pós-Graduação Lato Sensu em Gerência de Redes de Computadores e Tecnologia Internet do Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro – NCE/UFRJ.

Aprovada em Outubro de 2008.



Prof. Moacyr Henrique Cruz de Azevedo, M.Sc., UFRJ, Brasil

RESUMO

COELHO, Edinilson Machado. **REDES P2P: ANALISE DE APLICATIVOS** P2P. Monografia (Especialização em Gerência de Redes e Tecnologia Internet). Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro. Rio de Janeiro, 2008.

O interesse no compartilhamento de dados vem crescendo muito desde o surgimento da Internet e um dos principais motivos para isso tem sido a facilidade de se obter esses dados na rede pelo aumento da oferta, pela melhoria da velocidade de transmissão de dados, pelo aperfeiçoamento de técnicas de *downloads* e pelo surgimento de redes estritamente desenvolvidas para essa finalidade.

Nesse último aspecto se enquadra o tema desse trabalho, as redes Peer-to-Peer, ou redes ponto-a-ponto ou, somente, P2P e analise de seus aplicativos.

A idéia é ter computadores conectados à rede (chamados de nós, ou pares da rede) atuando como clientes ou como servidores dependendo apenas do fato de se estar disponibilizando (servindo) um dado na rede (*upload*) ou obtendo um dado da rede (*download*); podendo esse dado ser um arquivo ou mensagem.

Esses tipos de redes foram criados inicialmente no intuito de se compartilhar arquivos de música MP3 mas o seu poder de utilização vai muito além disso. Hoje, já existem aplicações na área de computação distribuída - software de busca de vida extraterrestre do SETI [SETI INSTITUTE 2005], gerenciamento de redes ópticas - projeto GigaMan [GRANVILLE 2005] e alguns projetos de aplicações educacionais colaborativas [RIBEIRO 2003] [ROCHA 2004].

Este trabalho não pretende esgotar o tema e nem fazer uma análise minuciosa dessa ou daquela tecnologia de rede, apenas despertar ainda mais o interesse para um tema atual e analisar seus aplicativos de forma a contribuir como fonte para um eventual estudo.

Palavras-Chave: Peer-to-Peer, Napster, Gnutella, FreeNet, eDonkey, ponto-a-ponto

ABSTRACT

COELHO, Edinilson Machado. **REDES P2P: ANALISE DE APLICATIVOS** P2P. Monografia (Especialização em Gerência de Redes e Tecnologia Internet). Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro. Rio de Janeiro, 2008.

The interest in the sharing of data is growing a lot from the appearance of the Internet and one of the main reasons for that has been the easiness of obtaining those data in the net for the increase of the offer, for the improvement of the data transmission speed, for the improvement of techniques of download and for the appearance of nets strictly developed for that purpose.

In that last aspect the theme of that work is framed, the nets Peer-to-Peer, or nets point-to-point or, only, P2P and analyze of their applications.

The idea is to have connected computers to the net (calls of us, or stop of the net) acting as customers or as servants just depending on the fact of her to be making available (serving) a die in the net (uploading) or obtaining a die of the net (download); being able to that given to be a file or message.

Those types of nets were created initially in the intention of sharing music files MP3 but his/her use power goes very besides. Today, already applications exist in the area of distributed computation - software of search of extraterrestrial life of SETI [SETI INSTITUTE 2005], administration of optical nets - I project GigaMan [GRANVILLE 2005] and some projects of applications education colaborativas [RIBEIRO 2003] [ROCHA 2004].

This work doesn't intend to drain the theme and nor to do a meticulous analysis of that or of that net technology, just to wake up still more the interest for a current theme and to analyze their form applications to contribute as source for an eventual study.

Word-key: Peer-to-Peer, Napster, Gnutella, FreeNet, Emulate, point-to-point

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AOL	- América Online
CPU	- Central Processing Unit
DHCP	- Dynamic Host Configuration Protocol
DNS	- Domain Name System
FTP	- File Transfer Protocol
HTTP	- Hyper Text Transfer Protocol
IP	- Internet Protocol
NAT	- Network Address Translation
PPP	- Point to Point Protocol
QoS	- Quality of Service
SLIP	- Serial Line Internet Protocol
TCP	- Transmission Control Protocol
UDP	- User Datagram Protocol
URL	- Uniform Resource Location
URN	- Uniform Resource Name
VPN	- Virtual Private Network
SETI	- Search for Extra-Terrestrial Intelligence

SUMÁRIO

1 INTRODUÇÃO

A Internet surgiu como meio de se compartilhar livre e incondicionalmente o conhecimento e os recursos disponibilizados em uma rede de dados, mas sempre se baseou na estrutura cliente-servidor o que torna os usuários dependentes de um núcleo que detenha e gerencie esse conhecimento ou recurso. É inquestionável o papel das redes peer-to-peer como solução para troca e compartilhamento de arquivos, bem como instrumento fundamental para implementação de sistemas distribuídos [ORAM 2001].

Existem várias definições para o modelo de comunicação P2P. o termo P2P refere-se a uma classe de sistemas e aplicações que empregam recursos distribuídos para executar uma função de forma descentralizada [MILOJICIC 2002].

Muito se tem falado atualmente deste modelo de interligação de redes de computadores, dando a idéia para muitos que este modelo é uma tecnologia recente. A área de Sistemas Distribuídos (*Distributed Systems* – DS) [COULOURIS 2000] já utiliza este modelo de pontos (também chamados nós) altamente acoplados, para compartilhamento de recursos computacionais como forma de aumentar o desempenho na resolução de problemas complexos. Dois computadores ligados um ao outro também é um exemplo muito simples de rede P2P. A nomenclatura P2P é recente, o modelo ponto-a-ponto não.

Os avanços em redes P2P e a rápida adoção pelos usuários conduziram ao desafio de se implementar soluções que suportem o compartilhamento de diferentes tipos de recursos e que sejam capazes de oferecer serviços mais complexos. Grande parte da mídia em torno do modelo P2P se deve ao fato do sucesso de *softwares* para compartilhamento de arquivos através da Internet que utilizam o modelo P2P de comunicação. Com a rápida expansão da Internet e com o custo cada vez menor das redes de alta velocidade, os usuários de redes de computadores têm cada vez mais acesso a uma gigantesca quantidade de informação. Essa informação pode ser acessada em tempo real, como é o caso das rádios de música que existem na Internet, entretanto grande parte desta informação ainda é fornecida na forma de arquivos. Por isso é necessário um serviço de rede que se encarregue de encontrar a informação na forma de arquivos. Os sistemas de compartilhamento de arquivos via Internet utilizando o modelo P2P surgiram para resolver este problema.

Um sistema de compartilhamento de arquivos que utiliza o modelo P2P consiste basicamente de um número variável de nós (*hosts* da rede) que

disponibilizam uma quantidade limitada de seus arquivos para outros nós da rede. Qualquer nó da rede é acessível por todos os nós de forma direta. A comunicação direta entre os nós da rede tem a vantagem de evitar possíveis pontos de gargalo (*bottleneck*) que são comuns em sistemas centralizados.

Uma característica comum a todos os sistemas P2P para compartilhamento de arquivos é a existência de um modelo de comunicação de rede que, no nível de aplicação, define uma rede virtual com seus próprios mecanismos de roteamento de dados [RIPEANU 2002].

A topologia desta rede virtual e os mecanismos de roteamento de dados definem de forma significativa propriedades da aplicação tais como: desempenho, segurança e anonimato do usuário. A topologia também define os custos em termos de recursos operacionais.

Outra característica marcante é que estes sistemas são formados por uma configuração de ambiente definidos por uma arquitetura distribuída, diferentes de outras formas de compartilhamento de arquivos, que são definidos por uma arquitetura centralizada.

O conjunto de características de sistemas que utilizam o modelo P2P para compartilhamento de arquivos aponta para o desenvolvimento destes sistemas com o auxílio de recursos da Inteligência Artificial. A forma cooperativa com que cada nó do sistema se comporta e as interações entre cada nó através de negociações, concessões, propostas e consultas, pode ser modelada através de uma abordagem de desenvolvimento de software orientado a agentes [BIGUS 2001].

Esse trabalho, em seu capítulo 2, procura abordar essas características de uma maneira geral e, no capítulo 3, apresenta analise dos aplicativos, escolhidos (Napster, Gnutella e FreeNet, eMule) por terem sido marcos dentro das gerações que se seguiram.

No capítulo 4, por fim, é feito um comparativo onde são mostrados aspectos de destaque das arquiteturas dentro das gerações de redes peer-to-peer. Apesar das arquiteturas serem enquadradas numa ou noutra geração, isso não quer dizer que a anterior tenha caído em desuso ou teve descontinuidade, muito pelo contrário, esses modelos sempre foram e continuam sendo estudados, aprimorados e amplamente utilizados.

Finalmente, no capítulo 5 é apresentada a conclusão do trabalho.

2 CONCEITUAÇÃO

De acordo com o Webopedia (www.webopedia.com) redes Peer-to-Peer significam:

"Um tipo de rede de computadores onde cada estação possui capacidades e responsabilidades equivalentes. Isto difere da arquitetura cliente/servidor, no qual alguns computadores são dedicados a servirem dados a outros".

O termo "peer-to-peer" refere-se a uma arquitetura de rede distribuída onde os participantes compartilham seus próprios recursos de hardware (banda de rede, processador) no intuito de prover os serviços de compartilhamento de arquivos e/ou de processamento oferecidos pela rede de forma descentralizada [SILVA 2003] [WIKIPEDIA 2005].

Estas definições capturam o significado tradicional da rede Peer-to-Peer.

Uma boa proposta técnica para P2P tem sido feita por Dave Winer da UserLand Software. Dave sugeriu que os sistemas P2P envolvam essas sete características chave:

- Interface de troca de arquivos fora do navegador de Internet.
- Computadores podem servir tanto como servidores como clientes
- Sistemas fáceis de usar e bem integrados.
- Sistema deve incluir ferramentas que ajudam usuários que queiram criar ou adicionar alguma funcionalidade.
- Sistemas que promovam conexão entre usuários.
- Sistemas que façam algo novo ou excitante .
- Sistemas que atendam a protocolos "cross-network" como SOAP ou XML-RPC.

O SETI com seu software que busca formas de vida extraterrestres é um exemplo de rede que provê compartilhamento de processamento. No compartilhamento de arquivos existe grande número de softwares de uso consagrado na Internet tais como Kazaa, LimeWire, eMule e outros. O projeto JXTA, por sua vez, é exemplo de plataforma de serviços.

O software Napster [NAPSTER 2005] trouxe popularidade para os programas de troca de arquivos usando o modelo de redes ponto-a-ponto (figura 2.1), embora esse seja um conceito antigo já utilizado, por exemplo, em serviços de telefonia (conversa simultânea), em jogos de interação direta (jogadas simultâneas) ou em aplicativos de FTP – File Transfer Protocol [ARAÚJO 2003].

A grande diferença entre o modelo P2P e o modelo cliente-servidor (figura 2.2) é que com a inexistência de um servidor central, é possível cooperar para a formação de uma rede P2P sem qualquer investimento adicional em hardware, não necessitando de computadores de alto desempenho para a utilização como servidores.

Redes Peer-to-Peer são uma alternativa para modelos centralizados onde há tipicamente um servidor e muitos clientes distribuídos.

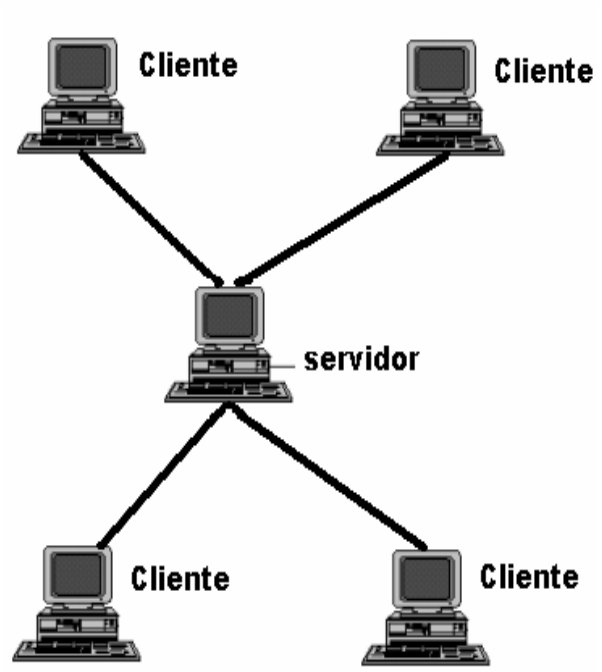


Figura 2.1: Modelo Ponto-a-Ponto

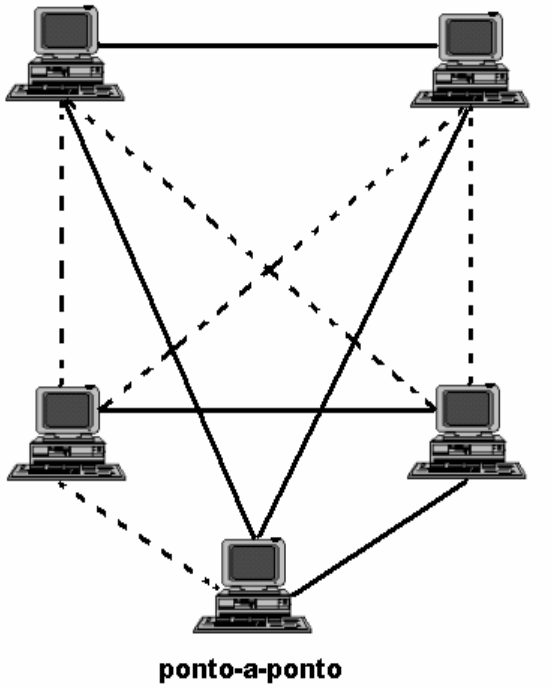


Figura 2.2: Modelo Cliente-Servidor

2.1 ARQUITETURAS DE REDES DE COMPARTILHAMENTO DE ARQUIVOS

O modelo P2P consiste na interconexão direta entre duas máquinas, visando o compartilhamento de arquivos e/ou serviços. Esta é uma definição clássica [ORAM 2001] que estipula que os nós da rede devem estabelecer suas conexões de forma independente e descentralizada, cada um deles desempenhando as funções de cliente e servidor. A grande vantagem do modelo P2P em relação ao modelo cliente-servidor é a pouca dependência de um servidor para centralizar conexões. Na verdade, alguns modelos P2P não utilizam nenhum tipo de servidor. Diversas topologias de rede podem ser utilizadas para interligar sistemas P2P. Por exemplo, podemos citar a centralizada, descentralizada e híbrida [WIKIPEDIA 2005].

2.1.1 Redes P2P Centralizadas

Este tipo de topologia de rede não faz uso de uma rede totalmente distribuída e, portanto, este tipo de sistema não é um sistema totalmente P2P.

Quando o usuário se conecta na rede, ele na verdade está se conectando a um ou mais servidores. Estes servidores servem como um índice para busca dos dados disponíveis, ou seja, quando um usuário se conecta informa ao servidor todos os dados que está dispondo na rede e o servidor atualiza os seus índices para que fiquem disponíveis a consulta para os outros usuários.

Quando um dos usuários deseja fazer uma busca por um determinado item, ele contacta um dos servidores e busca a informação desejada. A resposta do servidor é o endereço IP de uma das máquinas que tem o dado desejado (essa máquina obtida na pesquisa também já passou por todo o processo de conexão descrito no parágrafo anterior). Deste instante em diante as duas máquinas passam a se falar diretamente para efetuar a troca de dados. A figura 2.3 ilustra este processo. [CAMARGO 2004].

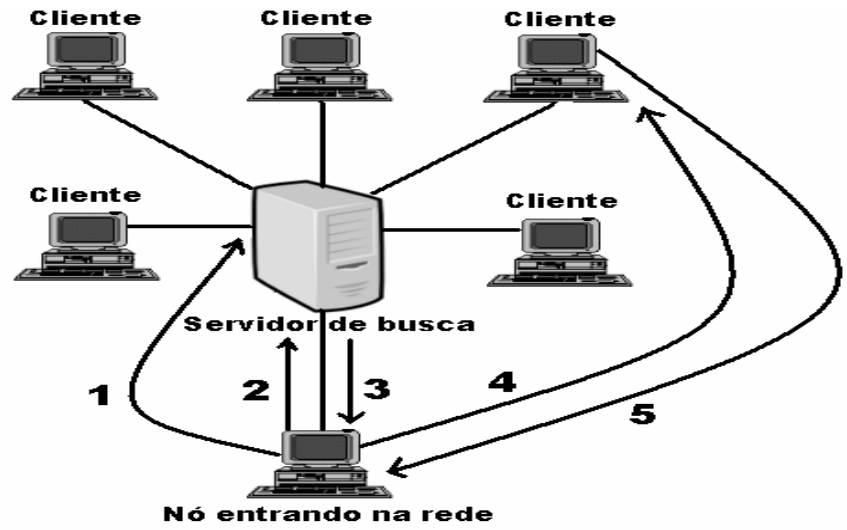


Figura 2.3: Funcionamento de uma Rede Centralizada.

O funcionamento de uma rede centralizada consiste em: 1. O novo cliente se conecta ao servidor e envia uma lista dos seus dados compartilhados. 2. O cliente já registrado no servidor solicita a pesquisa de dados. 3. O servidor retorna uma lista com o endereço de todos os nós que contém a informação desejada. 4. O cliente contacta o nó que contém o arquivo desejado e o requisita. 5. O nó contactado transfere o arquivo para o nó que o requisitou.

O serviço de localização centralizado traz vantagens e desvantagens. Um dos grandes ganhos é a garantia de que se o dado está disponível em algum dos nós da rede então ele será encontrado. Esta garantia não está presente em todos os sistemas P2P, como veremos adiante. Além disto, buscas com palavras chave e múltiplos critérios são facilmente implementadas. A desvantagem, entretanto, é a alta suscetibilidade a falhas pois existem pontos centrais de falha. Se estes pontos falharem a rede toda falha pois o servidor não hospeda nenhum arquivo, apenas a informação de quem os detém. Este tipo de estrutura é usado pelo Napster. [CAMARGO 2004].

2.1.2 Redes P2P Descentralizadas

Na topologia de redes P2P descentralizada, figura 2.4, também chamada de pura, cada ponto da conexão estabelece sua própria política de conexão, diferente do modelo cliente-servidor onde o servidor dita as regras. Cada ponto pode fazer requisições de serviço para o outro ponto, transferir dados de um para outro ponto, ou seja, cada ponto funciona como cliente e servidor. São redes 100% P2P. Estas redes são totalmente distribuídas e, portanto, não possuem um ponto central de falha. A única maneira de desmantelar a rede seria acabar com todos os seus nós, uma tarefa que é extremamente difícil. A maneira pela qual os nós se ligam à rede não é estruturada. Um nó que deseja entrar na rede simplesmente descobre alguns outros nós já presentes na rede (por broadcast ou qualquer outro meio) e se conecta a eles.

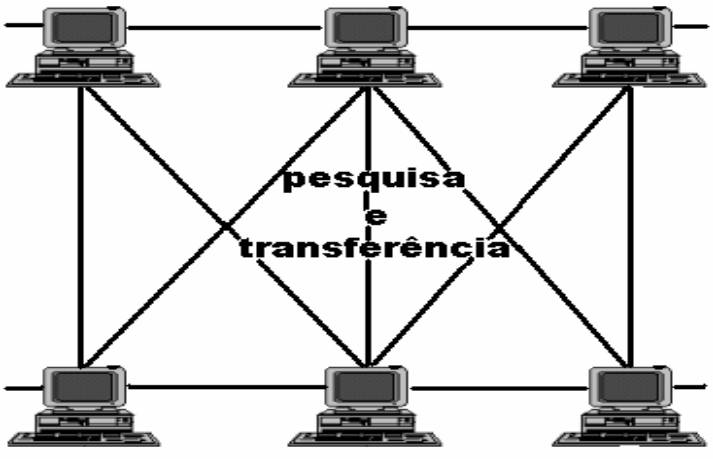


Figura 2.4: Rede P2P Descentralizada

Uma requisição nesse tipo de rede é propagada aos nós vizinhos até que se encontre o dado solicitado ou que se receba uma negativa na pesquisa. Essa

estrutura é utilizada, por exemplo, em redes Gnutella [GNUTELLA 2005] e em redes FreeNet [FREENET 2005].

Nessa arquitetura a busca pela rede é baseada em *hashing* - DIHA (Distributed Indexing with Hashing Architecture), ou em inundação ou *flooding* - DIFA (Distributed Indexing with Flooding Architecture). Ambas se caracterizam pela completa descentralização de seu funcionamento. Na arquitetura DIFA os mecanismos de busca e manutenção da infra-estrutura estão distribuídos pela rede. Neste tipo de sistema, cada nó é responsável por manter a listagem com os nomes dos seus próprios arquivos e responder quando receber um pedido de busca.

A rede deste tipo mais conhecida e estudada é a rede Gnutella. As redes deste tipo possuem uma limitação muito grande que está ligada ao fato de seu mecanismo de busca utilizar o mecanismo de inundação. Para que uma rede não fique saturada com a repetição infinita do *flooding* de mensagens, estas mensagens possuem um número máximo de nós que a mesma pode atravessar. Isto traz séria implicação, uma vez que, apesar de um arquivo existir no sistema e de que o nó que o contenha esteja on-line, uma busca para este arquivo pode falhar. Esse tipo de rede não oferece garantia alguma quanto à acessibilidade dos dados (o nó que contém o dado pode estar mais longe que o TTL – ex. requisição de N_q por um dado que está em N_{ar} na figura 2.5). Embora completamente descentralizadas, estas redes possuem deficiências sérias quanto à performance.

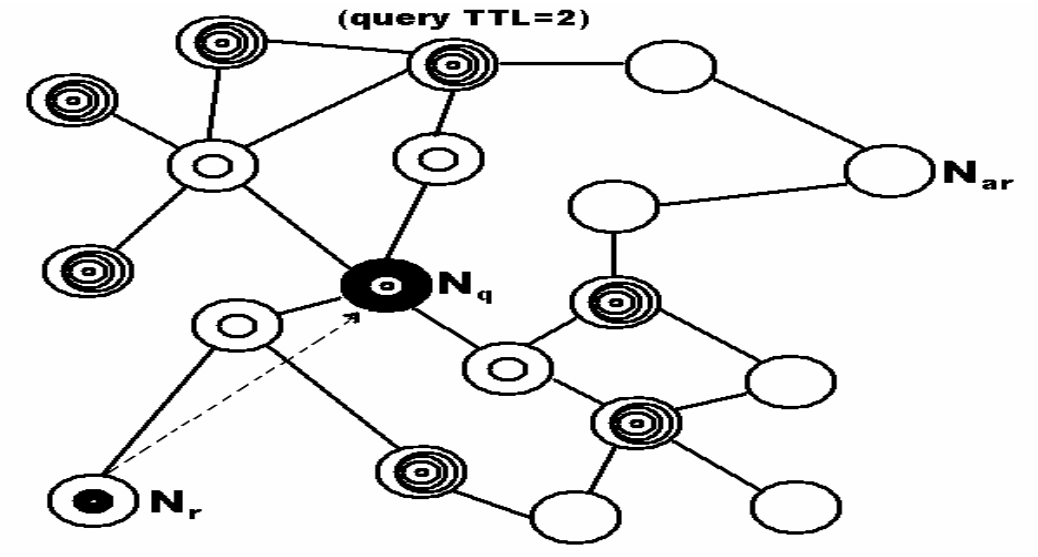


Figura 2.5: Representação - Funcionamento de uma rede P2P baseada em TTL

Uma representação do funcionamento de uma rede P2P baseada em TTL. O nó N_q transmite uma consulta requisitando o valor de uma chave que está localizada em N_r. Os círculos concêntricos representam o número de saltos da mensagem. [CAMARGO 2004].

A principal diferença entre as redes DIFA e DIHA está no mecanismo de busca. Nos sistemas com inundação cada ponto da rede é responsável pelo espaço de índices relativo aos arquivos que ele próprio contém. A arquitetura DIHA muda este conceito pois cada nó é responsável por um subconjunto do espaço total de índices. Quando o nó entra na rede recebe um espaço do conjunto dos índices dos arquivos, e ao sair a rede deverá designar estes índices para outro nó.

As buscas não são difundidas na rede sem direção como no flooding. Ao invés disto são direcionadas para o nó correto que é o responsável pelo respectivo índice dentro do espaço de índices. Os protocolos que implementam este tipo de arquitetura são bem mais complexos e existem poucos estudos sobre sua utilização.

2.1.3 Redes P2P Híbridas

A terceira arquitetura, representada pela figura 2.6, é a rede P2P híbrida ou CIA (Centralized Indexing Architecture), uma combinação das outras duas. Nessa estrutura alguns nós são designados como supernós. Os pontos clientes se conectam a um único supernó. Os supernós agem como os servidores centrais das redes P2P centralizadas. Cada supernó é conectado a um conjunto de outros supernós como os clientes de uma rede P2P descentralizada. Essa é a estrutura utilizada por softwares como o Kazaa, o iMesh e o eMule.

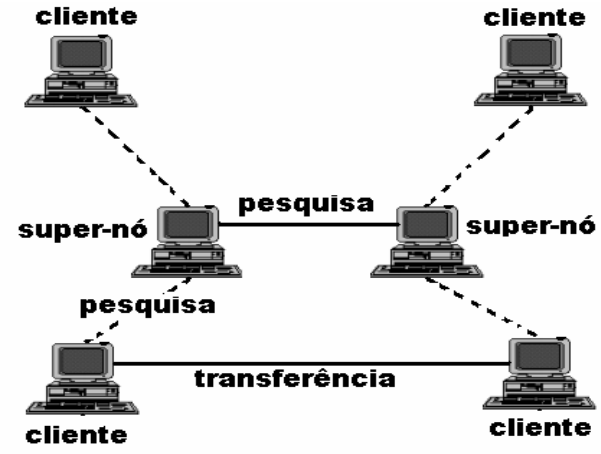


Figura 2.6: Rede P2P Híbrida

2.1.4 Arquiteturas baseadas em redes de superposição

Um problema muito comum em aplicações P2P é como localizar eficientemente um nó que contém uma informação desejada. Vimos em 2.1.1 e 2.1.2 algumas maneiras de resolver este problema. A desvantagem do método apresentado em 2.1.1 é a sua grande suscetibilidade a ataques enquanto que a desvantagem apresentada em 2.1.2 é a falta de escalabilidade e a falta de garantias quanto a acessibilidade aos dados da rede. As redes de superposição (overlay networks) foram criadas para tentar preencher as lacunas deixadas pelas outras arquiteturas.

Redes de superposição como CAN [RATNASAMY 2001], Chord [STOICA 2001], Pastry [ROWSTRON 2001], Kademlia [PETAR 2002] e Viceroy [DAHLIA 2002] criam uma topologia virtual sobre a topologia física da rede. A conexão entre os nós da rede Gnutella é feita de maneira arbitrária. Isto não acontece com redes de superposição (o protocolo Chord, por exemplo, utiliza o endereço IP das máquinas participantes para organizar a rede).

As redes de superposição têm as seguintes características:

- Interface de troca de arquivos fora do navegador de Internet.
- Garantia de acessibilidade dos dados
- Garantias para limites superiores para o tempo de procura (geralmente proporciona log (n) onde n é o número de nós da rede)
- Balanceamento automático de carga
- Auto organização

Essas características merecem uma melhor explicação. A conexão entre os nós da rede é feita com base no conteúdo de cada nó, ou seja, as conexões entre cada um dos nós podem ser expressadas em termos de funções matemáticas. Isso possibilita que a pesquisa por dados seja modelada através de um processo iterativo determinístico ao invés de uma busca baseada em inundação (por profundidade ou largura) no grafo de conexões. A análise deste processo permite, portanto, que sejam definidos os limites superiores para o tempo gasto em um processo de busca além da garantia de acessibilidade dos dados. Em termos abstratos podemos olhar uma rede de superposição como uma tabela de hash distribuída que suporta as operações de inserção, remoção e consulta de chaves. Tais chaves são, tipicamente, obtidas através de alguma função segura de hash (tal como o SHA-1). Um exemplo concreto deste esquema pode ser visto no exemplo mostrado na figura 2.7.

Todas as redes de superposição citadas definem algoritmos de estabilização para tratar a entrada e a saída (sendo por falha ou não) de nós da rede. A responsabilidade destes algoritmos é manter a topologia (definida pelas funções matemáticas de cada uma das redes) intacta. Essas redes são muito úteis para a criação de aplicações que requerem buscas rápidas, escaláveis, confiáveis e auto-organizadas por pares únicos de chave-valor.

Exemplo 1 - Chord O protocolo Chord define apenas uma operação: dada uma chave mapeá-la para um dado nó. O Chord, na verdade, é um arcabouço que aplicações P2P podem utilizar para efetuar as buscas e estruturar a rede.

Cada um dos nós possui uma identificação (ID). A identificação de um dado nó é obtida através da aplicação de uma função de hash no seu endereço IP. A rede é estruturada em forma de anel. Os nós são colocados neste anel em ordem crescente de ID elevado à potência 4. Cada nó i guarda uma tabela de apontadores para outros (log n) nós, onde n é o número de nós da rede. Esta tabela guarda os nós vizinhos de i. São eles: $i + 2^0$, $i + 2^1$, $i + 2^2$, $i + 2^3$ . . . $i + 2^m$, onde m é o número de bits máximo que o ID pode assumir (logo 2^m define o numero máximo de nós que a rede suporta). Formalmente os vizinhos do nó i são $(i + 2^{k-1}) \bmod 2^m$, $1 < k < m$. A figura 2.7 mostra um exemplo de vizinhança.

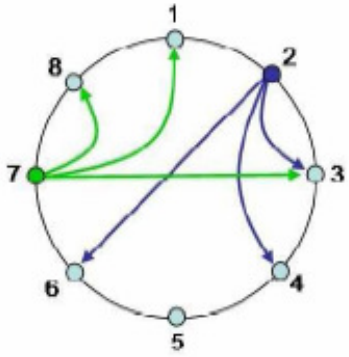


Figura 2.7 Vizinhança dos nós

As setas azuis apontam para os vizinhos do nó 2. As setas verdes apontam para os vizinhos do nó 7. Neste caso, m = 3, então, o número máximo de nós (n) é 2^3 = 8. Logo, cada nó tem apenas log(n) = m = 3 vizinhos na sua tabela de apontadores.

A busca pelo nó X com o identificador ID(X) é muito simples. Primeiramente a tabela de apontadores é investigada para encontrar o nó Y tal que ID(Y) é máximo e

que ID(Y) ≤ ID(X). Se ID(X) = ID(Y), o nó foi encontrado. Senão uma requisição é enviada para o nó Y. O nó Y repete os mesmos passos feitos acima. Até encontrar o nó desejado. Como a distância entre os apontadores cresce exponencialmente, em no máximo log(n) passos o nó procurado vai ser encontrado ou será determinada a sua inexistência na rede. As figuras 2.8 e 2.9 ilustram este processo.

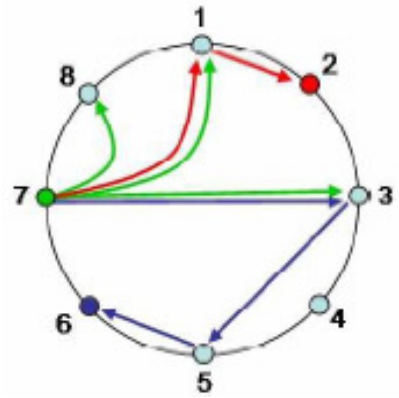


Figura 2.8 Buscas com sucesso por um nó

As setas verdes apontam para os vizinhos do nó 7. Em vermelho está indicado o caminho percorrido pelo processo para achar o nó com identificador 2 e em azul o caminho para se encontrar o nó com identificador 6.

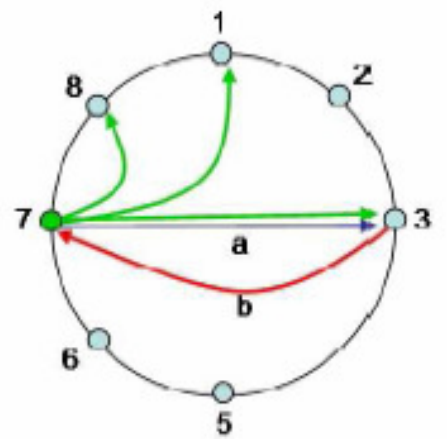


Figura 2.9 Buscas sem sucesso

As setas verdes apontam para os vizinhos do nó 7. Nó 7 tenta localizar o nó 4. O nó 4 não está na lista de apontadores do nó 7 e a pesquisa é repassada para o nó 3 (a). O menor apontador do nó 3 aponta para o nó 5 que é maior que o nó

procurado. É devolvido como resposta (b) o maior nó que é menor que o nó procurado, no caso o nó 3.

A localização dos dados no Chord é extremamente simples e por vezes até se confunde com o conceito de localização de um nó visto anteriormente.

Para cada dado disponibilizado na rede é calculado um ID. O cálculo deste ID é feito utilizando-se a mesma função de hash utilizada para calcular cada um dos IDs dos nós (portanto o espaço de IDs dos nós é o mesmo espaço dos IDs dos dados). Os dados disponíveis na rede, juntamente com a sua localização (tipicamente os endereços IP dos nós que o possuem), são guardados em uma estrutura semelhante a uma tabela de hash distribuída.

Esta tabela é implementada da seguinte maneira: um nó é responsável por guardar uma lista dos arquivos cujos IDs são maiores ou iguais ao seu próprio ID e que são menores que o ID do primeiro nó da sua lista de apontadores (ou seja, ele guarda todas as localizações dos arquivos cujos IDs estão entre o seu próprio ID e o ID de seu primeiro vizinho). A figura 2.10 ilustra o processo de busca por dados em uma rede com 6 nós com o espaço de IDs de 3 bits (m = 3).

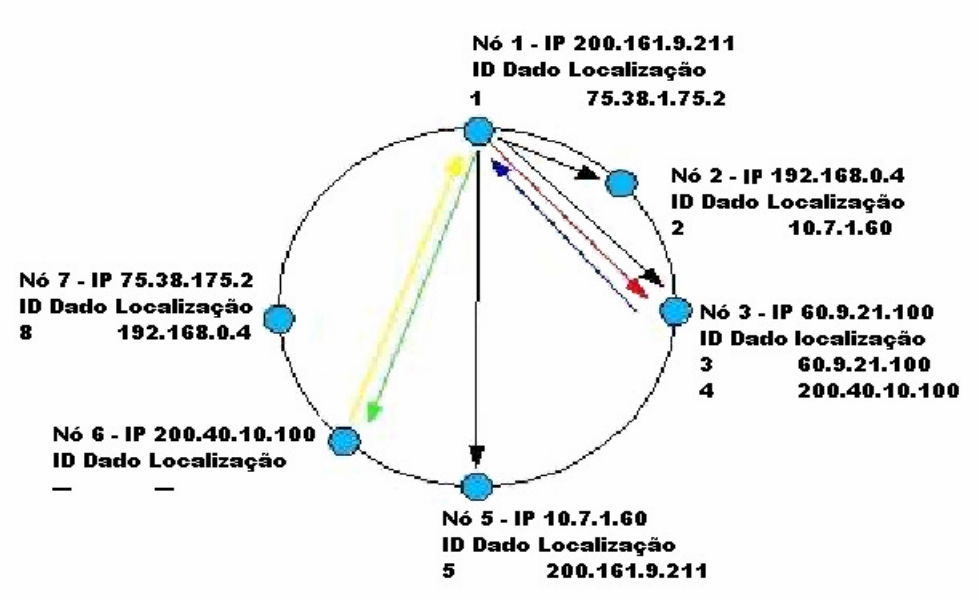


Figura 2.10: Busca por um dado na rede

Em cada um dos nós está indicado o seu identificador seguido pelo seu endereço IP (fictício). Logo abaixo destas informações está a parte da tabela de hash distribuída que compete a este nó. O nó 3, por exemplo, guarda as

informações da localização dos arquivos com ID 3 e 4, já o nó 6 não tem informação nenhuma a respeito da localização de qualquer dado. A figura representa a busca efetuada pelo nó com ID 1 por um dado com ID 4. As setas negras indicam os vizinhos do nó 1. O processo de busca se dá da seguinte maneira:

a) O nó 1 localiza na sua lista de apontadores o nó cujo ID é o mais próximo do ID procurado (4), neste caso o nó 3.

b) O nó 1 envia uma requisição de busca ao nó 3 (seta vermelha).

c) O nó 3 verifica que ele é o responsável pelo dado de ID 4 e que ele tem a sua localização o nó 3 envia uma mensagem de volta ao nó 1 com o endereço do nó que possui o dado desejado (seta azul).

d) O nó 1 com posse do endereço do nó 6 faz uma conexão direta e requisita o envio da informação (seta verde).

e) O nó 6 responde a requisição enviando o dado requisitado (seta amarela).

O Chord garante que os resultados da pesquisa são corretos ainda que apenas uma entrada da sua tabela de apontadores esteja correta. O desempenho do sistema degrada conforme o número de entradas incorretas na tabela de apontadores cresce. Como a tabela de apontadores é bem pequena (tipicamente log n entradas) é muito fácil mantê-la atualizada através de algoritmos de estabilização. É o que o Chord faz.

Testes mostraram que o Chord foi capaz de manter o tempo de pesquisa em log n, mesmo com 50% de probabilidade de falhas por nó [DIEGO 2003].

O Chord, além de rodar os algoritmos de estabilização a intervalos regulares de tempo, também funciona com múltiplos nós virtuais por nó real. Isto ajuda no balanceamento de carga dos nós pois propicia uma distribuição mais uniforme na rede [DIEGO 2003].

2.2 CLASSIFICAÇÃO DE APLICAÇÕES PEER-TO-PEER

Redes Peer-to-Peer são empregadas em diversas categorias de aplicações:

• **Comunicação e Colaboração**: Esta categoria inclui sistemas que fornecem infra estrutura de facilitação direta, normalmente tempo real, em comunicação e colaboração entre nós de computador. Exemplos incluem conversa e aplicações de mensagem imediatas (Chat e Irc) e transmissão de mensagens imediata (Aol, Icq, Msn);

• **Computação Distribuída**: Esta categoria inclui sistemas cujo objetivo é tirar proveito da capacidade de processamento (ciclos de CPU) de um computador ocioso na rede. Para isto é feito a decomposição da tarefa de processamento em pequenas unidades de trabalho para serem distribuídas aos diversos computadores da rede que executam sua parte correspondente e retornam os resultados ao sistema de coordenação centralizada responsável por dividir, distribuir as tarefas e reunir os resultados. Exemplos de tais sistemas incluem projetos como Seti@home, genome@home, e outros;

• **Suporte de Serviço de Internet**: Um número de aplicações diferentes baseadas em estruturas peer-to-peer surgiram para dar apoio a vários serviços de Internet. Exemplos de tais aplicações incluem sistemas peer-to-peer multicast, infra estruturas de vias indiretas de Internet e segurança de aplicações, fornecendo proteção contra negativa de serviço ou ataques de vírus;

• **Sistemas de Banco de Dados**: Trabalho considerável foi feito no desenho de sistemas de banco de dados distribuídos baseados em estruturas peer-to-peer. Existe a proposta de um Modelo Relacional Local (LRM) [ANDROUTSELLIS-THEOTOKIS 2004] onde o conjunto de todos os dados armazenados em uma rede peer-to-peer é assumido e compreendido por bancos de dados relacionais interligados que detêm o conhecimento e definem regras de tradução e dependências semânticas entre eles;

• **Distribuição de Conteúdo**: A maior parte dos atuais sistemas peer-to-peer estão incluídos na categoria de distribuição de conteúdo, que inclui sistemas e infra estruturas projetadas para o compartilhamento de mídia digital e outros dados entre usuários. Um sistema peer-to-peer de distribuição de conteúdo cria um meio de armazenamento distribuído que leva em conta a publicação, procura e recuperação de arquivos pelos membros da sua rede. Como os sistemas ficam cada vez mais sofisticados, algumas características não-funcionais devem ser implementadas, como segurança, anônima, aumento da escalabilidade e performance, bem como gerência de recurso e capacidade de organização.

2.3 CARACTERÍSTICA DAS REDES PEER-TO-PEER

2.3.1 Descentralização

Modelos P2P são contrários à filosofia de armazenagem e acesso de conteúdos em servidores centralizados. Em modelos tradicionais cliente-servidor a

informação é concentrada em servidores e distribuída através da rede aos computadores clientes. Sistemas centralizados são ideais para algumas aplicações e tarefas como as de gerenciamento de segurança. No entanto, ao contrário do que acontece em redes P2P centralizadas, as transferências não se dão diretamente entre nós-clientes. Sob esse aspecto, a topologia cliente-servidor, em comparação às redes P2P centralizadas, apresenta ineficiência pois produz enfileiramento de requisições em redes e desperdício de recursos de hardware; além de serem financeiramente onerosas por exigirem servidores robustos.

O ponto forte da descentralização é a distribuição dos dados ou dos recursos desejados. Por outro lado, em um sistema totalmente descentralizado cada ponto da rede é apenas mais um participante da rede disponibilizando seus dados e hardware, ou seja, não existe nenhum nó com a "visão" geral dos outros nós restantes ou dos arquivos que essa mesma rede disponibiliza. Esta é a principal razão pela preferência por redes P2P híbridas onde existe a informação centralizada da localização dos dados, embora um cliente apenas recupere esse dado diretamente de outro cliente.

Em sistemas totalmente descentralizados até a conexão com a rede é difícil. Um cliente precisa necessariamente saber o endereço de algum nó da rede ou se utilizar de lista de IP's conhecidos de nós da rede para estabelecer conexão. Ao se conectar com ao menos um nó atualmente conectado à rede, ele passa a conseguir descobrir outros nós em cadeia.

2.3.2 Escalabilidade

A escalabilidade é um benefício decorrente da descentralização, uma vez que é responsável pelo controle do crescimento do número de usuários e equipamentos conectados à rede, além da capacidade da rede, aplicações e processamento [ROCHA 2004].

Essa característica está diretamente relacionada com a comunicação entre os nós da rede P2P e com a descentralização que desobriga a existência de um servidor.

A escalabilidade é limitada por fatores relacionados à quantidade de operações centralizadas, como coordenação e sincronização, estados que precisam ser mantidos, paralelismo inerente das aplicações, e modelo de programação utilizado na construção do sistema.

Em geral, sistemas P2P tendem a ter maior escalabilidade que sistemas que utilizam o modelo cliente-servidor, pois, em um sistema cliente-servidor, os servidores são os únicos responsáveis por toda a carga do sistema. Ao contrário, em um sistema P2P, quando o número de clientes na rede aumenta, cresce também o número de servidores, uma vez que todos podem atuar como clientes e servidores, aumentando na mesma proporção a quantidade de recursos compartilhados. Dessa forma o sistema aumenta de tamanho sem perder escalabilidade.

2.3.3 Anonimato

O anonimato em redes Peer-to-Peer pode ser aplicado a toda forma de uso da rede pelos clientes. O anonimato pode proteger:

• **Autoria**: O autor ou o criador de um documento não pode ser identificado;

• **Publicação**: A pessoa que publicou um documento no sistema não pode ser identificada;

• **Acesso**: As pessoas que lêem ou consomem dados não podem ser identificadas;

• **Disponibilização**: Um documento recuperado de um cliente da rede não pode conter informações que identifiquem esse cliente;

• **Documento**: Servidores desconhecem os documentos que armazenam;

• **Pesquisa**: Um servidor não pode dizer que documento está sendo usado para atender a uma determinada solicitação.

Existem três tipos de anonimidade: o sender anonymity, relacionado ao remetente do dado; o receiver anonymity, que diz respeito ao destinatário do dado; e o mutual anonymity, que se refere a ambos.

Para que uma rede P2P garanta esse anonimato, ela pode utilizar técnicas conhecidas:

• **Multicasting ou brodcasting:** É usado para reforçar a anonimidade do receptor. Pode-se formar um grupo de clientes multicasting que desejem manter um dado anônimo. Um cliente interessado em determinado documento envia a requisição para esse grupo. Da mesma forma um cliente que possui o documento o envia para o mesmo grupo que o disponibiliza para o interessado. Dessa forma, um nó da rede que detinha a informação a envia sem saber quem a requisitou;

• **Troca do endereço IP (Spoofing):** Para protocolos sem conexão como o UDP, o anonimato pode ser reforçado pela troca do endereço IP do emissor da

mensagem. Isto nem sempre pode ser utilizado pois além de exigir troca de protocolo, muitos servidores de acesso rejeitam pacotes originados de IP's inválidos;

• **Troca de identidade**: Além da troca de IP do emissor, o anonimato pode ser assegurado trocando-se a identidade do cliente que responde a uma requisição na rede;

• **Dissimulação de rota**: Ao invés de realizar uma comunicação direta, quaisquer dois nós da rede podem se comunicar percorrendo alguns outros nós intermediários. Com essa técnica um nó que deseje esconder sua identidade, prepara caminhos secretos com outros nós, ficando o detentor do dado no fim do caminho. Variando-se diferentes caminhos, com diferentes números de nós alternados com freqüência, pode-se aumentar significativamente o grau de anonimidade na rede;

• **Utilização de pseudônimos**: Um servidor proxy gera pseudônimos não roteáveis para os clientes de uma rede. Um cliente pode abrir uma "conta" e ser reconhecido pela rede sobre o retorno recebido pela abertura dessa conta enquanto esconde sua verdadeira identidade. Essa técnica assegura anonimidade do emissor e se baseia em confiabilidade ao servidor proxy;

• **Localização involuntária**: Um cliente pode receber e armazenar de forma involuntária um documento de outro cliente. Por ser involuntário, o anfitrião não pode responder sobre possuir tal documento.

2.3.4 Auto-Organização

Em cibernética, auto-organização é definida como "um processo onde a organização de um sistema aumenta espontaneamente sem que este aumento esteja sendo controlado pelo ambiente ou um sistema externo" [HEYLIGHEN 1997].

Em sistemas P2P, a auto-organização é necessária para prover características como: escalabilidade, resistência às falhas, conexão intermitente dos recursos, e custo de propriedade. Os sistemas P2P podem ter dimensão imprevisível em termos do número dos aplicativos ou do número dos usuários. É muito difícil prever qualquer um deles, o que requer constante re-configuração do sistema centralizado. Um significativo aumento da dimensão da rede resulta em um provável aumento das falhas, que irá requerer a auto-manutenção e o auto-reparo dos sistemas. Raciocínio similar aplica-se à desconexão intermitente: é difícil para qualquer configuração predefinida permanecer intacta num período de tempo longo.

A adaptação é requerida para assegurar as mudanças causadas pelos clientes que, a todo o momento, se conectam e se desconectam dos sistemas P2P.

Finalmente, por ser muito caro ter equipamento dedicado e/ou pessoas para controlar um ambiente tão flutuante, a gerência é distribuída entre os nós.

2.3.5 Custo de Propriedade (Posse)

Uma das premissas da arquitetura P2P é o compartilhamento. O compartilhamento reduz o custo de possuir todo o sistema e o custo de mantê-los. Isto é aplicável a todas as classes de sistemas P2P e em computação distribuída. A eliminação de computadores centralizados para armazenar a informação traz a conseqüente redução do custo da propriedade e de manutenção da informação.

2.3.6 Conectividade Ad-Hoc

Conectividade ad-hoc diz respeito à alta rotatividade dos participantes de uma rede P2P. Por causa dessa rotatividade, o paralelismo das aplicações distribuídas da rede não se dá em todos os sistemas o tempo todo pois alguns aplicativos estarão disponíveis todo o tempo, alguns estarão parcialmente disponíveis, e alguns não estarão disponíveis. Sistemas P2P e aplicações em computação distribuída precisam estar cientes desta natureza "temporária" para poder assegurar o funcionamento dos aplicativos da rede. Devido à natureza dos sistemas P2P, essa temporariedade, ao contrário do que acontece em sistemas distribuídos tradicionais, é muito comum.

2.3.7 Performance

A performance é fator de grande preocupação em sistemas P2P. A melhoria no desempenho desses sistemas é buscada aliando-se a alta capacidade de armazenamento distribuída (por exemplo, Napster, Gnutella) à quantidade de ciclos de execução (por exemplo, SETI@home) dos dispositivos espalhados através de uma rede. Por causa da natureza descentralizada destes modelos, o desempenho é influenciado por três tipos de recursos: processamento, armazenamento, e banda de rede. A largura da banda é fator principal quando um grande número de mensagens é propagado na rede e uma quantidade grande de arquivos está sendo transferida entre diversos nós da rede. Isto limita o escalabilidade do sistema. O desempenho neste contexto não se mede apenas em termos de milisegundos, mas no tempo que

se leva para receber um arquivo ou de quanto da largura de banda da rede se consome em uma requisição.

Em sistemas centralmente coordenados, essa coordenação entre os nós da rede é controlada e mediada por um servidor central, embora os nós também possam mais tarde se conectar diretamente. Isto faz com que estes sistemas fiquem vulneráveis aos mesmos problemas que enfrentam servidores centralizados. Para superar as limitações de um coordenador centralizado, arquiteturas P2P híbridas propõem distribuir a funcionalidade do coordenador entre múltiplos usuários-servidores que cooperam para satisfazer as requisições de usuários. O DNS é um exemplo de um sistema P2P hierárquico que melhora o desempenho definindo uma árvore dos coordenadores, com cada coordenador responsável por um grupo de nós da rede. Uma comunicação entre nós em grupos diferentes é conseguida através de um coordenador de um nível mais elevado.

Em sistemas de coordenação descentralizada, tais como Gnutella e FreeNet, a comunicação é assegurada individualmente por cada participante da rede. Tipicamente usam mecanismo de repasse da mensagem na procura pela informação e pelos dados. O problema com tais sistemas é que terminam emitindo um grande número de mensagens replicando-a de um nó a outro. Cada ponto da rede contribui para o aumento na largura de banda da rede e no tempo requerido para obter os resultados de uma requisição. A largura de banda para uma requisição é proporcional ao número das mensagens emitidas, que é por sua vez proporcional ao número dos nós que devem processar o pedido antes que os dados sejam encontrados.

2.3.8 Segurança

Além das necessidades típicas de segurança das redes baseadas no modelo cliente-servidor como a confiabilidade nos nós e objetos compartilhados, esquemas de troca de chaves criptográficas de sessão, criptografia, certificado e assinatura digital, os modelos P2P trouxeram novas necessidades:

• **Codificação multi-chave**: alguns sistemas de compartilhamento de arquivo buscam proteger tanto o objeto compartilhado quanto a identidade do autor desse objeto, a identidade do usuário que requisitou o objeto, e a identidade do usuário que disponibilizou esse mesmo objeto na rede utilizando-se de esquema baseado

em mecanismo de criptografia assimétrica com uso de chave pública e chaves confidenciais múltiplas;

• **Isolamento de código**: Alguns sistemas P2P de computação distribuída, em sua filosofia, necessitam efetuar download do código de aplicação em máquinas nós da rede. Diante disso, se torna crucial a proteção dessas máquinas de códigos potencialmente maliciosos e, por outro lado, proteger, também, o código de máquinas maliciosas. Proteger uma máquina significa:

1 - Reforçar a segurança para que códigos externos não consigam trazer danos para ela ou que seus dados somente sejam acessados em modo-seguro. Para isso se utilizam técnicas que isolam o código externo ao seu próprio domínio de proteção;

2 - Reforçar propriedades de segurança para impedir que dados sensíveis escapem para máquinas maliciosas. Aqui se utilizam técnicas da teoria do fluxo de informação e verificação de modelo;

• **Gerência de direitos digitais**: Sistemas de compartilhamento de arquivos usando redes P2P facilitam demasiadamente a cópia. É necessário poder proteger os autores do roubo de sua propriedade intelectual. Uma das maneiras de se assegurar a propriedade seria adicionar aos arquivos uma assinatura que os tornassem reconhecíveis, mas de um modo que não afetasse o conteúdo, funcionando como se fosse uma espécie de “marca d´água” nos moldes das utilizadas em imagens;

• **Reputação e responsabilidade**: Em sistemas P2P é muito importante manter a reputação dos nós para impedir que os mal-comportados prejudiquem o sistema inteiro. A reputação requer maneiras de se medir o quão "bom" ou "útil" um nó é. Por exemplo, devem existir regras para que se um determinado usuário compartilhar grande número de arquivos interessantes, sua reputação seja elevada. Freeloader é um termo que designa usuário que baixa arquivos de sistemas P2P sem oferecer outros em troca. Um freeloader tem geralmente uma reputação baixa. Para impedir este tipo do comportamento não-cooperativo, alguns mecanismos de contabilidade necessitam ser planejados. Os sistemas atuais confiam basicamente na avaliação cruzada feita entre uma comunidade de usuários autenticados. Entretanto, a própria autenticação não fornece nenhuma garantia do comportamento de um par;

• **Firewall**: As aplicações P2P requerem conexões diretas entre os nós da rede. Entretanto, em ambientes corporativos as redes internas são isoladas da rede externa (a Internet), reduzindo o direito de acesso às aplicações. Por exemplo, a maioria dos firewalls bloqueia mensagens TCP recebidas. Isto significa que uma máquina dentro de um firewall não estará acessível a uma máquina externa à rede. Usuários maliciosos usam freqüentemente IP mascarados ou a tecnologia (NAT) de tradução de endereço da rede para compartilhar uma conexão de Internet entre diversas máquinas, que os leva ao mesmo problema de inacessibilidade. Entretanto, o acesso enviado através da porta 80 (HTTP) é freqüentemente permitido pelos firewalls, então, alguns mecanismos foram planejados para permitir conexões entre máquinas escondidas por firewall ou NAT e máquinas na Internet. Isto tem suas limitações porque requer que a conexão seja iniciada da máquina escondida. Quando ambos os nós que querem se comunicar residem atrás de firewalls diferentes, o problema torna-se ainda mais difícil de ser contornado pois requer um servidor central na Internet que forneça uma conexão entre os nós escondidos.

2.3.9 Transparência e Usabilidade

Em sistemas distribuídos, transparência é tradicionalmente associada com a habilidade de se conectar localmente aos diversos sistemas disponibilizados pela rede. A forma mais comum de transparência é conhecida como ‘transparência de localização’, mas existem outras formas de transparência como de acesso, de falha, de mobilidade, de concorrência e etc. [COULOURIS 94]. Com o passar do tempo algumas dessas transparências, como por exemplo a por falha foram sendo classificadas para que fossem levadas em conta no desenvolvimento de sistemas distribuídos.

No começo, a Internet deu atenção particular à transparência no nível do protocolo (TCP/IP), chamado assim 'transparência end-to-end de endereço'. O termo 'end-to-end' significa que determinadas funções em uma comunicação entre duas entidades podem ser executadas somente com o conhecimento e o estado mantidos nestas entidades no nível da aplicação. Isto implica que o estado de uma comunicação da aplicação não está mantido na rede, mas nos pontos de extremidade de uma comunicação e que algum ponto na rede sabe o nome e o endereço do outro ponto de comunicação, o que nem sempre é verdade. O escasso número de endereços IP, aliado a introdução das Intranets e de usuários móveis,

resultou em números de IP que são válidos somente durante uma única sessão. Os exemplos incluem SLIP, PPP, VPNs, uso de firewalls, DHCP, endereços confidenciais, tradutores de endereço de rede (NATs), DNS, etc.

Isto teve implicações significativas para o P2P e foi também uma das razões para a introdução do P2P. Os sistemas P2P surgiram com esquemas diferentes do usado pelo DNS para tradução de IP's.

A tradução de IPs em nomes WEB não traz uma completa transparência. URLs são intensamente utilizadas no lugar de URNs que eram, supostamente, quem disponibilizaria a transparência de nomes. Ao lado da transparência de nomes e endereços em P2P, existe também a necessidade de "transparência de administração" visto que usuários são tipicamente inexperientes e não podem, portanto, administrar seus softwares e dispositivos.

Um aplicativo P2P não deve requerer nenhum ajuste especial no software ou configuração de redes ou de dispositivos a fim de poder funcionar. É desejável, também, que seu software seja auto-atualizável. Além disso, as requisições de dispositivos e de rede para o funcionamento de aplicações P2P devem ser transparentes aos usuários. Devem trabalhar na Internet, nas Intranets, e nas redes confidenciais, usando banda larga ou ligações dial-up. Devem também ser transparentes quanto ao canal utilizado, ou seja, devem aceitar trabalhar com toda variedade de dispositivos, tais como palms, desktops, celulares, etc.

Outra forma de transparência está relacionada à segurança e à mobilidade. A autenticação pode ser significativamente simplificada se for feita de forma automática e transparente aos usuários e proxies. Usuários móveis devem ser suportados, isto é, deve-se permitir aos usuários trabalhar independentemente de onde e como estão conectados à Internet ou intranets.

Um nó pode utilizar aplicações P2P das seguintes formas:

- Como usuário de serviços, através de interfaces WEB;
- Envolvido em uma aplicação não-P2P, tipicamente uma plataforma (ex: .NET);
- Como um software P2P instalado (ex: Napster).

2.3.10 Resistência a Falhas

Um dos principais objetivos de projetos de sistema P2P é evitar um ponto central da falha. Embora a maioria dos sistemas P2P "puros" já faça isto, eles constantemente se deparam com falhas de desconexão, problemas de partição e

falhas nos nós. Estas falhas são mais pronunciadas em algumas redes (por exemplo, wireless) do que outras. Seria desejável continuar a colaboração ativa entre os nós ainda conectados na presença de tais falhas. Um exemplo disso seria uma aplicação, tal como o genome@home, que divide o processamento entre os nós.

A desconexão de um nó da rede pode trazer como resultado a indisponibilidade dos recursos. Isto pode ocorrer devido a uma falha da rede ou porque o nó que foi desconectado era o que hospedava o recurso desejado. A replicação de recursos cruciais ao funcionamento da rede ajuda a aliviar o problema. As redes P2P tais como Napster e Gnutella representam os sistemas que têm um mecanismo passivo e descontrolado de replicação baseado unicamente na popularidade do arquivo. Dependendo da aplicação que funciona sobre estas redes, pode ser necessário fornecer determinadas garantias de persistência. Isto requer uma política mais forte e maior confiança em replicação.

Os sistemas de publicação anônimos, tais como o FreeNet e o Publius, asseguram a disponibilidade por replicação controlada. A aplicação Oceanstore mantém duas camadas hierárquicas das réplicas e, através de monitoração de domínios administrativos, evita emitir réplicas aos pontos da rede com alta probabilidade de estarem correlacionadas às falhas. Entretanto, como um recurso no sistema P2P pode ser mais do que um simples arquivo (tal como um proxy para a Internet, compartilhamento de espaço de armazenamento, ou o poder de computação compartilhada) os conceitos de sistemas de arquivos replicados têm que ser estendidos a esses tipos adicionais de recursos. As soluções de grid de computação fornecem resistência às falhas do nó reiniciando o processamento em nós diferentes.

Um aspecto inovador de sistemas P2P é que a responsabilidade da manutenção do sistema está completamente distribuída e necessita ser endereçada a cada nó para assegurar a disponibilidade. Isto é completamente diferente dos sistemas cliente-servidor, onde a disponibilidade é uma responsabilidade do servidor.

2.3.11 Interoperabilidade

Embora existam muitos sistemas P2P ainda não há suporte para permitir interoperabilidade entre eles. A introdução de diferentes plataformas, diferentes

sistemas e diferentes aplicações trabalhando juntas numa dada infra estrutura, abre uma série de questões a respeito de segurança que acabam por afetar a interoperabilidade. Quanto maiores forem as diferenças dentro de uma infra estrutura, maiores serão os problemas de interoperabilidade associados.

Existem alguns pré-requisitos para haver interoperabilidade:

• Como determinar se um sistema pode interoperar;

• Como fazer os sistemas comunicarem-se (protocolo, etc.);

• Como fazer os sistemas trocarem pesquisas e dados;

• Como determinar se os sistemas são compatíveis nos níveis mais elevados do protocolo (ex: um sistema pode 'contar' com a propriedade de busca do outro);

• Como os sistemas publicam e mantêm o mesmo nível de segurança, QoS e confiança.

No passado, haviam maneiras diferentes de se chegar à interoperabilidade, tal como a definição de padrões (por exemplo, padrões de IEEE para Ethernet, token ring, e wireless); as especificações comuns, (por exemplo, grupo de gerência de objeto); código de fonte comum, (por exemplo, OSF DCE); código aberto (por exemplo, Linux); e padrões de fato (por exemplo, Windows ou Java).

JXTA [JXTA 2007] Community Project é exemplo do esforço de se obter interoperabilidade em aplicativos de código aberto, tentando impor um padrão de fato. Nele existe uma arquitetura mínima suportada como base permitindo aos outros sistemas contribuírem nas partes que podem ser compatíveis com suas próprias implementações.

3 PRINCIPAIS IMPLEMENTAÇÕES

São identificadas três gerações de redes P2P [WIKIPEDIA 2005]. A primeira se caracterizava por centralizar nos servidores da rede P2P uma lista que relacionava determinado conteúdo da rede com o nó da rede que detinha esse dado. Isto trouxe problemas para alguns servidores que acabavam permitindo, mesmo que sem saber, acesso a dados por vezes cobertos por direito autoral (Ex: Napster).

Na segunda geração foi contornado o problema da hospedagem dessa lista e, ainda, o problema de escalabilidade, o que deu origem ao Gnutella.

A terceira geração implementou o conceito de anonimato dos clientes e dos servidores de arquivo (Ex: Freenet).

3.1 NAPSTER

Em maio de 1999, Shawn Fanning, um estudante universitário de 18 anos, criou um software que combinava o sistema de mensagens instantâneas do IRC, o sistema de compartilhamento de arquivos do Windows e Unix e as capacidades de busca das máquinas de pesquisa. O seu objetivo era criar um sistema que tornasse mais fácil compartilhar arquivos mp3. Ele batizou o seu programa de Napster (o seu apelido na universidade), fundou uma empresa com o mesmo nome e passou a distribuir o cliente de graça.

O Napster se tornou a aplicação de internet de crescimento mais rápido de todos os tempos, atingindo um pico de 13.6 milhões de usuários em fevereiro de 2001 (fonte: comScore Media Metrix), trouxe notoriedade para o modelo de aplicações P2P devido ao sucesso alcançado por permitir que qualquer cliente da rede Napster pudesse anunciar os arquivos MP3 que dispõe em sua máquina e baixar arquivos MP3 de outras máquinas conectadas ao servidor Napster.

A Napster devido à questões judiciais teve que suspender suas atividades no ano de 2002. Comprada pela Roxio nesse mesmo ano voltou à atividade com uma única diferença: já não é mais um sistema de conteúdo gratuito.

Napster é considerado um P2P híbrido pois em sua estrutura existe a figura de servidores centrais. Além da troca de arquivos MP3 ainda disponibiliza outros serviços como mensagens instantâneas, salas de bate-papo, grupos de amigos e informativos sobre músicas. A própria estrutura do MP3, com informações em metadados, trouxe eficiência no gerenciamento da troca de arquivos entre os nós pelo servidor.

O Napster utiliza um protocolo estilo HTTP e todos os nós da rede são simétricos, isto é, com capacidade de funcionar tanto como clientes quanto como servidores. Essa simetria é um atributo que distingue o sistema peer-to-peer de muitas arquiteturas de sistemas distribuídos convencionais.

Um grande grupo de servidores centrais dedicados mantém um índice dos arquivos que estão sendo atualmente compartilhados por nós ativos. Cada nó mantém uma conexão a um dos servidores centrais para onde é pedida a localização do arquivo solicitado. Os servidores então cooperam para processar a requisição e retornar uma lista com arquivos e localizações ao usuário solicitante. Recebendo os resultados o nó pode decidir iniciar uma troca de arquivo diretamente de outro nó.

Além de manutenção de um índice de arquivos compartilhados, os servidores centralizados também controlam o estado de cada nó no sistema, conservando informações dos metadados tais como a conexão informada dos nós, a largura de banda e a duração que o nó permaneceu conectado ao sistema. Este metadado é devolvido com os resultados de uma requisição para que o nó requisitante possa distinguir os possíveis sites para downloads.

O grupo de servidores Napster compõe-se de aproximadamente 160 servidores. Cada nó estabelece uma conexão com só um servidor. Quando um nó emite uma requisição, o servidor do nó primeiramente informa os arquivos compartilhados por “usuários locais” no mesmo servidor e, posteriormente, os arquivos que combinam com o requisitado e que são compartilhados por “usuários remotos” em outros servidores no grupo, como demonstrado na figura 3.1.

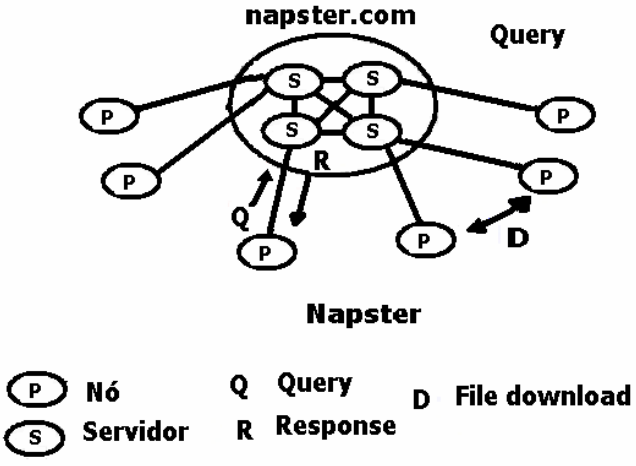


Figura 3.1: Modelo da rede Napster

Fonte: SAROIU, S.; GUMMADI, P. K.; GRIBBLE, S. D., 2001. p.3.

Toda mensagem dirigida de/para um servidor central Napster segue a seguinte estrutura [DING 2003]:

Quadro 1 - Estrutura Napster de Mensagens

Bytes	0-1	2-3	N bytes
Conteúdo	Length	Function	Payload

Onde:

Length: especifica o tamanho de “payload”;

Function: define o tipo de mensagem do pacote;

Payload: o pacote propriamente dito.

As mensagens permitidas nessa rede são:

• INITIALIZATION: Um cliente registrado Napster envia uma mensagem de login ao servidor.

Quadro 2 – Estrutura Napster (Initialization)

Nick	Password	Port	Client Info	Link Type

Onde:

Nick & Password: Identificação do usuário;

Port: Porta que o cliente “escuta” para transferência dos dados;

Cliente Info: String com a informação da versão do cliente;

Link Type: Inteiro com o código da banda de Internet do usuário, podendo ser: 0=desconhecido; 1=14.4Kbps; 2=28.8Kbps; 3=33.6Kbps; 4=56.7Kbps; 5=64k ISDN; 6=128k ISDN; 7=Cable; 8=DSL; 9=T1; 10=T3

• CLIENT NOTIFICATION OF SHARED FILE: O cliente envia, sucessivamente, todos os arquivos que deseja compartilhar.

Quadro 3 – Estrutura Napster (Client Notification Of Shared File)

Filename	MD5	Size	Bitrate	Frequency	Time

Onde:

Filename: Nome do arquivo;

MD5: Hash do arquivo compartilhado;

Size: Tamanho do arquivo em bytes;

Bitrate: Taxa de amostragem do arquivo MP3 em Kbps;

Frequency: Freqüência do arquivo MP3 em Hz.

Time: Duração da música em segundos.

• FILE REQUEST: O cliente de download emitirá inicialmente uma mensagem de busca ou visualização de arquivos. A primeira mensagem é do tipo “File Request”.

Quadro 4 – Estrutura Napster (File Request)

Artist name	Title	Brirate	Max results	Link Type	Frequency

Onde:

Artist name: Nome do artista da música que se deseja;

Title: Nome da música que se deseja;

Bitrate: Taxa de amostragem do arquivo MP3 que se deseja;

Max results: Máximo de resultados a serem retornados numa busca;

Link Type: Tipo de conexão;

Frequency: Freqüência em Hz do arquivo MP3 desejado.

• RESPONSE AND BROWSE RESPONSE: O servidor envia uma mensagem em resposta a mensagem de busca ou visualização do cliente.

Quadro 5 – Estrutura Napster (Response And Browse Response)

Filename	MDS	Size	Bitrate	Frequency	Length	Nick	IP	Link Type

Onde:

Filename: Nome do arquivo retornado;

MD5: Hash do arquivo retornado;

Size: Tamanho do arquivo retornado;

Bitrate: Taxa de amostragem do arquivo retornado;

Frequency: Freqüência do arquivo retornado;

Length: Tamanho do arquivo retornado, em bytes;

Nick: Usuário que está compartilhando o arquivo;

IP: 4 bytes que designam o endereço IP do usuário que detém o arquivo;

Link-Type: O mesmo do login.

• DOWNLOAD REQUEST: Para receber um arquivo da rede, uma requisição de download é enviada ao servidor. Na solicitação de um arquivo o cliente informa o nome do arquivo e o nome (apelido) do detentor do arquivo.

Quadro 6 – Estrutura Napster (Download Request)

Nick	Filename

• DOWNLOAD ACK: O servidor envia uma “resposta” com uma confirmação de download com maiores informações: velocidade da linha, número da porta, etc.

Quadro 7 – Estrutura Napster (Download Ack)

Nick	IP	Port	Filename	MD5	Link Type

• ALTERNATE DOWNLOAD REQUEST: Difere da mensagem DOWNLOAD REQUEST apenas no fato que o usuário que está compartilhando o arquivo só pode realizar saídas TCP porque está atrás de um firewall que bloqueia as mensagens de entrada. Essa mensagem deve ser usada para solicitar arquivos de usuários que especificaram que sua porta é ‘0’ na mensagem de login.

• ALTERNATE DOWNLOAD ACK: Enviada aos “uploaders” quando sua porta for especificada como ‘0’ indicando que estão atrás de um firewall e necessitam enviar os dados.

• FILE TRANSFER: Deste ponto em diante os nós não mais enviam mensagens para o servidor central. O nós que solicitou um arquivo realiza uma conexão TCP com a porta especificada na mensagem enviada pelo servidor. Na requisição do arquivo desejado é enviado uma mensagem HTTP (GET) com seguinte formato:

Quadro 8 – Estrutura Napster (File Transfer)

Nick	Filename	Offset

Onde:

Nick: Apelido do cliente;

Filename: Nome do arquivo desejado;

Offset: byte offset do arquivo para início da transmissão. Necessário para poder prosseguir uma transferência previamente interrompida. Uma vez iniciada a transferência, uma mensagem DOWNLOADING FILE é enviada ao servidor para notificação e, do mesmo modo, quando a transferência é completada, o cliente envia uma mensagem DOWNLOAD COMPLETE.

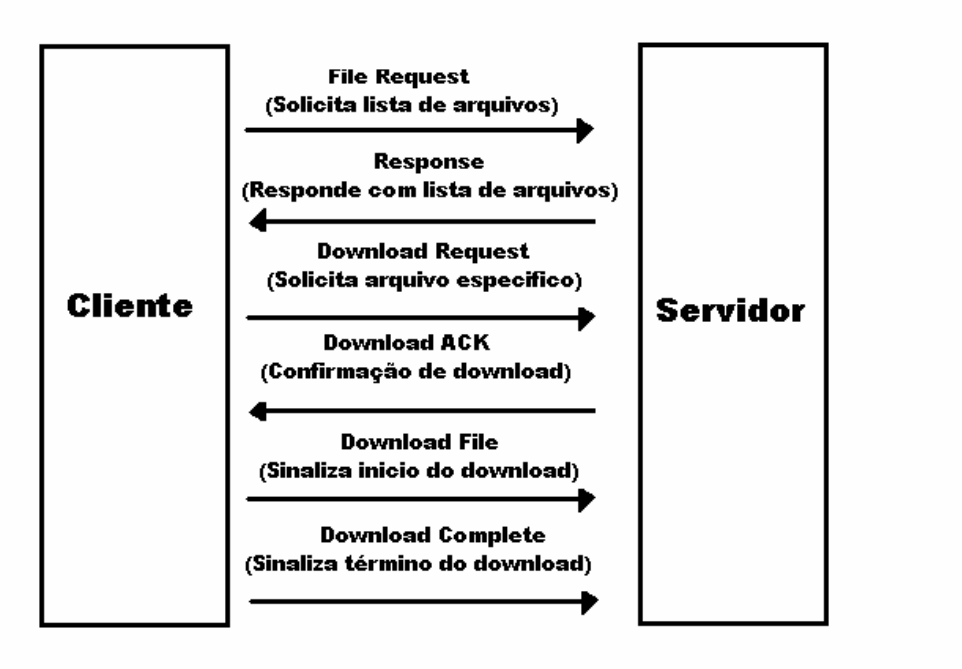


Figura 3.2: Seqüência de mensagens no Napster

Na figura 3.2 podemos observar o fluxo normal de mensagens da rede Napster. Numa busca de arquivo são trocadas as duas mensagens iniciais (File Request e Response). Se um arquivo da lista for o desejado, acontecem então as demais mensagens.

3.2 GNUTELLA

O Gnutella (neologismo proveniente da junção de GNU – de GNU General Public License – com Nutella, sobremesa da Ferrero feita de chocolate com avelãs)

surgiu em março de 2000, criado por Justin Frankel e Tom Pepper, da Gnullsoft, também criadores do Winamp, foi desenvolvido em apenas 14 dias como um experimento. Inicialmente tinha por objetivo compartilhar receitas culinárias. Porém, a AOL (America Online), que comprou a companhia, descontinuou o desenvolvimento do mesmo por seu envolvimento com a indústria fonográfica (o caso Napster já estava em discussão).

Foi então que desenvolvedores de software livre intervieram: Bryan Mayland realizou a engenharia reversa do programa e Ian Hall-Beyer e Nathan Moinvaziri criaram um site e um canal de IRC com toda a documentação necessária, com o objetivo de buscar novos desenvolvedores.

Desde então o Gnutella tem evoluído mais como uma rede do que um software propriamente, visto que existem atualmente vários softwares clientes que o utilizam, como o LimeWire e o Phex (com versões para Windows, Mac e Linux), o Morpheus e o BearShare (para Windows), o Mutella e o Qtella (para Linux). Só o Limewire contabiliza em média 1,5 milhão de usuários por dia (http://www.limewire.com/english/content/uastats.shtml).

O modelo de rede peer-to-peer Gnutella introduzido pela AOL em 2002 é, na verdade, um protocolo de compartilhamento de arquivos. O objetivo de Gnutella é fornecer solução para compartilhamento distribuído de arquivos. A natureza descentralizada de Gnutella fornece um bom nível de anonimidade para usuários, mas introduz também um grau de incerteza quanto à origem dos dados. Aplicações que implementam esse protocolo permitem aos usuários a busca e a baixa de arquivos de outros usuários conectados à Internet. Em uma rede Gnutella, cada nó mantém aberta uma conexão TCP (porta 6346 por padrão) com pelo menos um outro nó, criando assim a rede virtual de *servents* ao nível de aplicação. As requisições e as mensagens de manutenção de grupo são propagadas usando a técnica de inundação. Os *servents*, como são chamados os nós Gnutella pelos desenvolvedores, fornecem interfaces do lado cliente pelas quais os usuários podem emitir solicitações, pesquisar resultados, aceitar requisições de outro *servents*, verificar existência de dados solicitados contra o seu conjunto de dados local, e responder com resultados correspondentes. Esses nós são também responsáveis por controlar o tráfego em segundo plano.

Para que, inicialmente, um novo nó/*servent* se conecte à rede, ele une-se a um dos vários hosts conhecidos e que estão quase sempre disponíveis. Uma vez

conectado à rede (tendo uma ou várias conexões abertas com nós já na rede), os nós enviam mensagens para “interagir”. As mensagens (consultas) podem ser broadcasted (isto é, enviadas a todos os nós com os quais o remetente tem conexões TCP abertas) ou simplesmente retornadas no caminho inverso ao da consulta (back-propagation). Várias características do protocolo facilitam este mecanismo de broadcast/back-propagation.

Primeiro cada mensagem tem um identificador aleatório gerado; depois cada nó guarda uma pequena memória das mensagens recentemente roteadas, usadas para implementar o back-propagation e impedir retransmissão; e por último as mensagens são marcadas com o *time-to-live* (TTL), isto é, possuem tempo de vida para especificar o número máximo de vezes que a consulta pode se propagar para outros nós.

As consultas ocupam muita largura de banda da rede Gnutella, então, a especificação restringe o tamanho máximo dessas mensagens a 256 bytes. Os pacotes Gnutella são da forma [DURANTE 2004]:

Quadro 9 – Estrutura Gnutella de Mensagens

Bytes	0 - 15	16	17	18	19 - 22
Conteúdo	Message ID	Function ID	Time To Live	Hops	Payload length

Onde:

Message ID: em conjunto ao TCP/IP é usado unicamente para identificar uma transação;

Function ID: é utilizado como aviso (resposta), consulta (resposta), ou requisição;

Time To Live: é o tempo de vida do pacote, isto é, quantas vezes mais o pacote será encaminhado;

Hops: conta o número de vezes que um dado pacote é encaminhado;

Payload length: é o tamanho em bytes do corpo do pacote.

As mensagens permitidas nessa rede são:

• PING: Um nó que se junta à rede inicia transmissão de uma mensagem Ping para todos os nós conectados para anunciar sua presença. Esse tipo de mensagem

é enviado sem “payload”. Isto significa que o descritor de cabeçalho possui toda a informação necessária.

• PONG: O nó que recebe uma mensagem de Ping, envia-a para os nós vizinhos e retorna uma mensagem Pong em resposta. Na mensagem Pong existe informação a respeito do nó, tais como o seu endereço IP e o número e o tamanho de arquivos compartilhados. Para redução do tráfego da rede, os nós colocam as mensagens Pong em cache e as fornecem em resposta ao Ping quando possível.

Quadro 10 – Estrutura Gnutella (Ping / Pong)

Bytes	0 -1	2 – 5	6 – 9	10 – 13
Conteúdo	Port	IP Address	File Count	Total File Size

Onde:

Port: é a porta que o nó utiliza para se comunicar;

IP Address: o endereço IP do nó;

File Count: Um contador que possui o número de arquivos que o nó compartilha;

Total File Size: o tamanho total de todos os arquivos que o nó compartilha em kilobytes.

• QUERY / QUERY HIT: Mensagens QUERY (consultas) contêm string com as palavras chaves da busca. Cada requisição recebida é checada contra nomes de arquivos armazenados localmente. As mensagens são do tipo broadcast. As mensagens QUERY HIT são mensagens de resposta a uma mensagem QUERY adicionada de informação necessária para baixa do arquivo.

Quadro 11 – Estrutura Gnutella (Query)

Bytes	0 -1	2 – N
Conteúdo	Mininum Speed	Search Query Address

Onde:

Minimum Speed: Se um nó em particular quer se comunicar com um nó requisitor, ele deve ser apto a fazê-lo a uma velocidade mínima ou maior. Só os nós que se encaixam neste critério podem enviar uma resposta à consulta;

Search Query Address: Esta é uma string de dados de busca que contém a requisição do nó. Esta string deve conter um caractere vazio para marcar o fim do descritor.

Quadro 12 – Estrutura Gnutella (Query Hit)

Bytes	0	1 - 2	3 - 6	7 -10	11 - (N-1)	N-(N+16)
Conteúdo	Number of Hit	Port number	IP address	Speed	Result Set	Peer ID

Onde:

Number of hits: O número de arquivos que correspondem à string de busca. Os detalhes individuais de cada arquivo são agrupados;

Port Number: A porta na qual o nó fica escutando para estabelecer uma comunicação;

IP Address: O endereço do nó;

Speed: Velocidade de transferência em kilobytes por segundos;

Result set: Este campo possui os detalhes do arquivo. Ele manda o índice do arquivo armazenado no nó hospedeiro (este é um número dado a um arquivo pelo Gnutella), o tamanho do arquivo em bytes, e uma string com o nome do arquivo encontrado. Este campo é terminado por dois caracteres vazios. O índice e o tamanho do arquivo são ambos de quatro bytes de comprimento. A string para o nome do arquivo é de pelo menos um byte de comprimento;

Peer ID: Ele apenas identifica um nó na rede.

• GET / PUSH: As baixas de arquivos são feitas diretamente entre dois nós da rede usando-se esses tipos de mensagens.

Quadro 13 – Estrutura Gnutella (Get / Push)

Bytes	0 – 15	16 – 19	20 – 23	24 – 25
Conteúdo	Peer ID	File index	IP address	Speed

Onde:

Peer ID: Esta é a identidade única de um nó que possui o arquivo requisitado. Este campo é obtido do descritor QueryHit retornado por um determinado nó;

File index: Este é o mecanismo usado pelo nó requisitor para indicar qual arquivo ele quer do nó com o peer ID. Este índice é obtido pelo descritor Query Hit;

IP Address: É o endereço do nó hospedeiro que contém o arquivo a ser transferido;

Port: Número da porta pela qual o arquivo é transferido.

A figura 3.3: Demonstra graficamente a atividade dessas mensagens nas redes Gnutella. Nela podemos observar que mensagens Ping são responsáveis por 50% de todo o tráfego da rede.

Os usuários da rede Gnutella procuram por conteúdo emitindo consultas arbitrárias ao sistema. O nó que deseja começar a busca emite uma mensagem de consulta a todos os nós conectados diretamente a ele.

Cada um destes nós, por sua vez, replicará e emitirá a mesma consulta a seus próprios vizinhos, exceto ao nó que originou a pergunta. Este processo (flooding) se repete em cada nó dentro de um raio estabelecido.

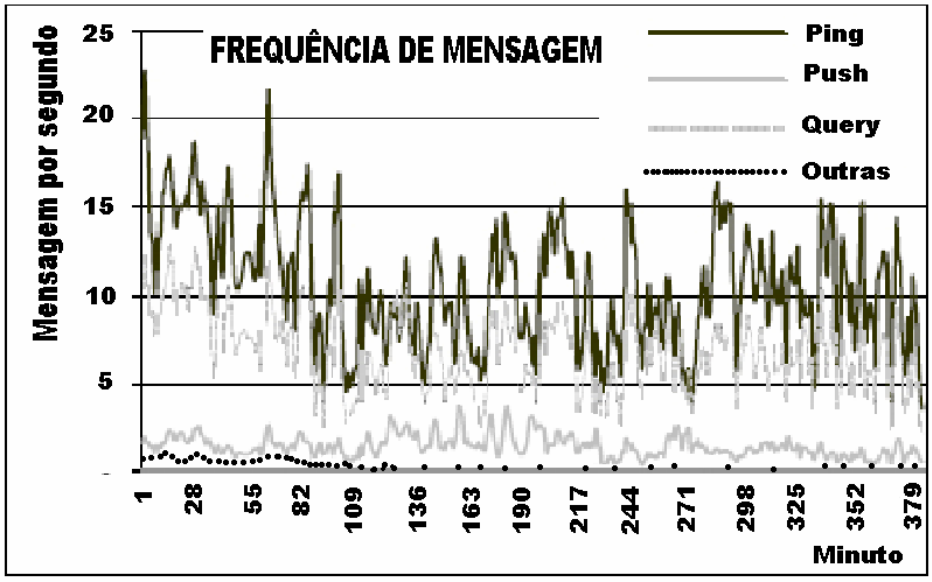


Figura 3.3: Freqüência de mensagens na rede Gnutella

Fonte: RIPEANU, M., 2002. p.2.

Medidas adicionais são adotadas para certificar-se de que essas consultas não se propaguem inútil e infinitamente na rede. As mensagens de Gnutella tem os valores do TTL (Time to Live), incluso na mensagem, decrementados em cada nó.

Os nós descartam as consultas quando o campo TTL chega a zero. Os nós são também responsáveis por descartar as consultas em duplicidade. As consultas incluem um "campo de velocidade mínima" dirigida aos nós para que respondam à consulta somente se puderem satisfazer à velocidade mínima de transferência de arquivos [BERKES 2003].

A mensagem poder conter, também, um "campo de critério de busca", permitindo que os clientes procurem por arquivos baseados em critérios, e, às vezes em clientes específicos. As consultas genéricas, que não possuem campo de critério, são baseadas em uma série de palavras-chave, para que sejam encontrados os arquivos que combinam com todas elas.

Adicionalmente, um tipo especial de consulta pode solicitar uma lista de todos os arquivos servidos por um nó.

Um nó responde a uma consulta quando o seu conteúdo satisfaz à consulta. A resposta do nó contém o seu endereço IP e uma porta onde o nó pode ser acessado para transferir o arquivo.

Uma vez que o nó que realizou a busca recebe uma resposta positiva, ele sabe o endereço IP do nó, ou dos nós, que atendeu, ou atenderam, à sua consulta. Sendo assim, ele abre uma conexão TCP/IP diretamente para o nó que possui o arquivos. Se o nó que contém o arquivo está embaixo de um firewall, o nó solicitante envia uma consulta Push. Esse tipo de consulta diz ao nó que contém o arquivo para enviá-lo diretamente ao para o solicitante. Caso as duas máquinas estejam em baixo de um firewall, outras técnicas mais elaboradas devem ser utilizadas.

O "Ultrapeer" é um conceito que não foi especificado no protocolo original de Gnutella, mas que tem um papel importante nesse modelo de rede. O esquema de "Ultrapeer" melhora a eficiência da rede e a escalabilidade categorizando nós em “clientes regulares" e "supernós". Um supernó é um host confiável, com grande largura de banda de rede e que pode agir como um proxy para um grande número de clientes, como exemplifica na figura 3.4. O supernó minimiza o fluxo de distribuição de mensagem extensiva de um cliente na rede. Este cliente pode ser um usuário de modem com baixa largura de banda de rede. Neste caso o usuário do modem (cliente regular) usa um supernó como um ponto de entrada na rede.

Como as mensagens da rede Gnutella são em broadcast, as consultas e as respostas a essas consultas se dariam de forma exponencial. Usado de forma hierárquica, o conceito de "Ultrapeer" deixa a rede Gnutella bem escalar, reduzindo

largamente o número dos nós realmente envolvidos na distribuição de mensagem, imitando o próprio conceito da Internet: os nós de baixa largura de banda são conectados aos routers maiores (os supernós) que transmitem a maioria dos dados em "backbones" de alta largura de banda.

O protocolo Gnutella por si só não garante a propriedade “resistência à falhas”, na verdade, sempre se espera que hajam nós suficientes conectados à rede no momento em que a pesquisa se propaga na rede para que essa consulta possa atingir o nó que possui o item pesquisado. De fato, estudos demonstraram que apenas uma pequena parte de usuários Gnutella permanecem online tempo suficiente para responder às pesquisas de outros nós. A figura 3.5 mostra o funcionamento de uma rede Gnutella.

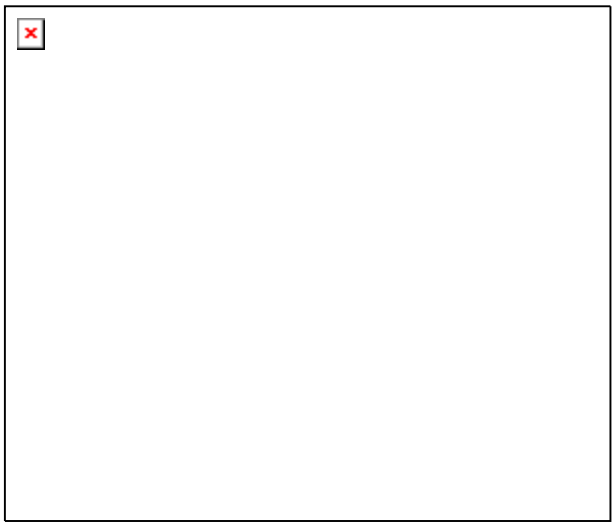


Figura 3.4: Ultrapeer (pontos cheios)

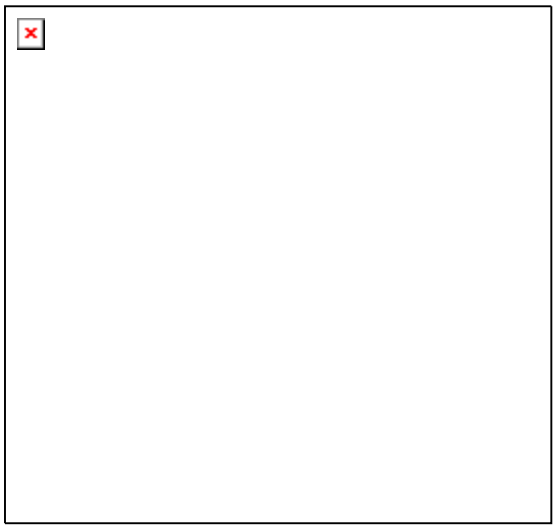


Figura 3.5: Modelo de Rede Gnutella

Fonte: BERKES,J. E.,2003 p.4 Fonte: SAROIU, S.,GUMMADI,P.K.; GRIBBLE, S. D.,2001. p.3.

3.3 FREENET

FreeNet é um sistema P2P completamente descentralizado e distribuído. Criado em 1999 por Ian Clark sua topologia se assemelha à do Gnutella exceto que no FreeNet, o objetivo principal foi operar como um sistema de arquivos distribuídos independente da localização entre vários computadores, permitindo que a inserção, armazenamento e requisição de dados se dê anonimamente. Foi desenvolvido para que, uma vez no sistema, o usuário tenha como requisitar arquivos sem que nenhum outro usuário possa determinar quem está fazendo tal requisição. Do mesmo modo, se um usuário disponibiliza um arquivo na rede, não deve haver maneira de se

determinar quem o fez. Finalmente, o operador de um nó FreeNet não deve poder ter informação sobre os dados que estão sendo armazenados localmente em disco.

Para o FreeNet, um nó é um simples computador que “percorre” todo o sistema, e todos os nós são tratados como iguais pela rede. Cada nó mantém sua própria unidade de armazenamento que é disponibilizada para a rede tanto para leitura e escrita, quanto para a tabela de roteamento dinâmico que contém endereços de outros nós. Assim, o FreeNet cria um enorme sistema de armazenamento distribuído e com grande redundância o que o torna fortemente tolerante a falhas, pois vários nós da rede podem fornecer os mesmos dados como mostra a figura 3.5. O sistema foi desenhado para se adaptar ao uso de padrões (patterns) movendo, replicando, e apagando arquivos, de forma transparente e de acordo com a necessidade, para fornecer um serviço eficiente sem recorrer a “broadcast” de busca ou a índices de localização centralizados.

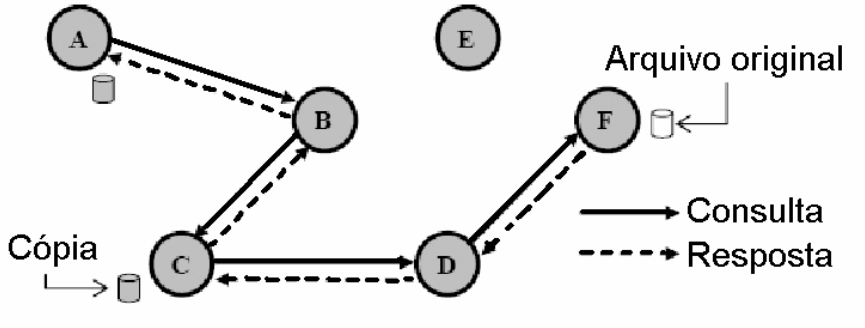


Figura 3.6: Mecanismo de replicação de arquivos do FreeNet
Fonte: ROCHA, J. et al., 2004. p.28.

Da mesma forma que no Gnutella, para se conectar a rede FreeNet tem que se conhecer previamente algum endereço de IP conhecido de algum nó já conectado para com ele estabelecer conexão TCP/IP.

No sistema FreeNet um usuário não pode acessar os dados que foram disponibilizados localmente em seus discos pois os mesmos são armazenados criptografados. Também não pode solicitar conteúdo de nós específicos, apenas enviam uma solicitação à rede e aguardam que algum nó responda com os dados desejados. Todo arquivo na rede FreeNet é identificado por chave gerada através da função hash SHA1 (utilizando chaves públicas e privadas). Assim, a pesquisa não se dá através de nomes de arquivos e sim de chaves previamente conhecidas. Além

disso, os dados fornecidos em cada nó são encriptados com uma chave que não fica disponível no nó, de forma que cada nó desconhece a natureza do conteúdo que está fornecendo para a rede. Esta é uma característica essencial de segurança do sistema [BERKES 2003].

O Freenet está implementado como uma rede P2P adaptativa de nós que se consultam para armazenar ou fornecer arquivos de dados, que são chamados de chaves de localização independente (location-independent keys). Existem três tipos básicos de chave [CLARK 2005]:

• **Keyword-Signed Key (KSK)** que é baseada numa pequena string descritiva escolhida pelo usuário quando este insere um arquivo. Esta string é basicamente misturada para fornecer a KSK. Para permitir que outros recuperem um documento, basta que estes forneçam esta chave;

• **Signed-Subspace Key (SSK)** que são usadas para identificar subespaços pessoais. Esta chave permite aos usuários construírem uma reputação pelo fornecimento de um documento enquanto permanecem anônimos, mas ainda identificáveis;

• **Content-Hash Keys (CHK)** que permite aos nós verificarem se um documento é uma cópia genuína do original. Ele também permite ao autor atualizar um documento se este utiliza um subespaço privado (com SSK).

No modelo básico, primeiramente uma chave é passada de nó a nó através de uma cadeia de requisições no qual cada nó estabelece um local de decisão sobre onde irá enviar a próxima requisição, no estilo do roteamento IP. Depois dependendo da chave requisitada a rota irá mudar. O algoritmo de roteamento se ajusta adaptativamente à rota de tempos em tempos para promover um desempenho eficiente. Cada nó possui apenas conhecimento a respeito de seus nós vizinhos, para manter a privacidade, então, para cada requisição é dado um tempo limite em “hops” (hops-to-live), que é decrementado a cada nó para evitar cadeias infinitas. Posteriormente cada requisição assume um identificador único, que faz com que os nós possam prevenir loops rejeitando requisições que eles já viram antes. Este processo continua até que a requisição seja satisfeita ou o tempo de “hops” (hops-to-live) exceda o limite estipulado. Então o sucesso ou a falha é retornado pela cadeia para informar ao nó de origem.

Dentre os problemas que podem afetar o desempenho da rede P2P descentralizada do Freenet, podemos citar a comunicação de rede. A velocidade de

conexão é menor do que a velocidade de E/S (entrada/saída) e de processamento, o que causa um gargalo na comunicação. Este problema é enfatizado pela grande natureza paralela do Freenet.

Outro problema é que, como não existe um armazenador central de índices, as mensagens devem passar por muitos “hops” para buscar através do sistema o arquivo desejado. Cada “hop” não só adiciona carga à banda total como também aumenta o tempo necessário para a realização da consulta. Se um nó for inalcançável pode levar vários minutos até que a comunicação seja interrompida.

Enquanto Gnutella conserva a largura de banda da rede transferindo arquivo para fora da rede, o FreeNet transfere os arquivos inteiramente dentro de sua rede. A razão para isso é estritamente de segurança, pois, por especificação, o FreeNet deve ocultar as fontes reais dos dados. Cada nó FreeNet conhece somente até seus nós de vizinhança, entretanto, os arquivos, na realidade, podem vir diretamente de um dos nós vizinhos ou de "n-hops" além de seus vizinhos. Não há nenhuma maneira de se saber isso ao certo. Arquivos grandes podem ser divididos em múltiplas partes sendo que cada parte do arquivo tem uma chave hash associada (CHK) que possa ser usada para verificar a integridade dos dados do arquivo [BERKES 2003].

Após transferir um arquivo, ou uma parte do arquivo, os nós receptores da mensagem aplicam a função hash no conteúdo do arquivo com SHA1 e se asseguram, de que o resultado combine o CHK original que já é conhecido, pois esta foi a "chave" usada na solicitação do arquivo. Assim, após ter executado um download e ter verificado que o hash confere, um usuário pode estar absolutamente certo que o arquivo recebido é uma cópia inalterada do arquivo solicitado.

FreeNet utiliza os seguintes tipos de mensagens, em cada uma é incluído o identificador do nó (para detecção de loop), um valor hops-to-live (similar ao TTL do Gnutella) e um identificar do nó de origem e destino:

• **Inserção de dados**: Um nó insere novos dados na rede. Uma chave e os dados reais (o arquivo) estão incluídos;

• **Pedido de dados**: Uma solicitação de um certo arquivo. A chave do arquivo solicitado é também incluída;

• **Resposta de dados**: Uma resposta iniciada quando o arquivo solicitado é localizado. O arquivo real é também incluído na mensagem de resposta;

• **Os dados falharam**: Um fracasso ao tentar localizar um arquivo. A posição (nó) do fracasso e a razão também estão incluídas.

3.4 EMULE

A idéia do projeto eMule surgiu no dia 13 de maio de 2002 quando um garoto chamado Merkur se reuniu com outros desenvolvedores para criar um aplicativo que utilizasse a rede eDonkey de maneira mais eficiente e mais agradável para o usuário. O eDonkey já era um sistema de distribuição de arquivos ponto-a-ponto popular mas que necessitava de alguns ajustes para ser um sucesso. Foi por essa necessidade que surgiu o eMule.

O eMule é um aplicativo ponto-a-ponto (*peer-to-peer*) de compartilhamento de arquivos baseado no protocolo eDonkey. Ele também trabalha com a rede eDonkey2000, que possui de 11,5 a 12,5 milhões de usuários conectados simultaneamente compartilhando cerca de 700 a 900 milhões de arquivos. É um programa de código aberto lançado sobre a GNU General Public License e roda no sistema operacional Windows possuindo uma versão para Linux denominada aMule. Além de utilizar a rede eDonkey, o eMule tem uma rede ainda experimental própria denominada Kademlia que adota uma política descentralizada - diferente da rede eDonkey.

A rede do eMule possui centenas de servidores e milhões de clientes. Estes devem se conectar ao servidor para ter acesso à rede. Tal conexão se estabelecerá até que o cliente saia do sistema. Os servidores realizam um serviço de indexação centralizado (assim como era no Napster) e não se comunicam com outros servidores.

A conexão com a rede eMule é feita, inicialmente, com uma conexão única entre cliente-servidor. Em seguida, o cliente irá se conectar a outros servidores e a outros diversos clientes expandindo sua ligação na rede por via tanto de conexões TCP como UDP.

Ao iniciar o aplicativo eMule, o cliente tentará se conectar a diversos servidores em paralelo, mas estabelecerá a conexão TCP apenas com um. O eMule não permite que um cliente se conecte a vários servidores ao mesmo tempo ou mude de servidor dinamicamente sem a intervenção do usuário. No estabelecimento da conexão (*handshake*) entre o cliente e o servidor, este último envia uma identificação de quatro bytes ao cliente que é denominada *Client ID*. Esta ID será

válida enquanto durar a conexão que somente será encerrada caso o cliente se desconecte daquele servidor, feche o eMule ou perca o acesso à internet.

O cliente poderá receber uma ID alta (*High ID*) ou baixa (*Low ID*). Ele terá uma ID baixa se ele não aceitar receber conexões, o que ocorrerá caso ele esteja usando algum tipo de firewall não configurado adequadamente para o eMule, estiver conectado através de uma NAT, por um servidor proxy, ou se o servidor estiver muito ocupado. Do contrário, ele receberá uma ID alta. Ter uma ID baixa significa ter um acesso restrito à rede e ser possivelmente rejeitado por servidores, que dão preferência a clientes com ID alta. Esta última, garante ao usuário total proveito da rede e será a mesma quando ele se conectar a outros servidores. A ID alta de um cliente só será alterada quando este mudar seu endereço IP, pois elas são calculadas da seguinte forma:

> "Assumindo que o IP do cliente seja, X.Y.Z.W , a ID alta será (X + 28)*(Y + 216)*(Z+224)*W (representação *big endian*). A ID baixa é sempre inferior a 16777216 (0x1000000), mas não existe um padrão de como é calculada. Vale lembrar que a ID baixa difere entre servidores." [Adaptado de: Yoram Kulbak and Danny Bickson, The eMule Protocol Specification, 2005].

Estabelecida a conexão, o cliente envia ao servidor sua lista de arquivos compartilhados. O servidor armazena essa lista na sua base de dados junto com outras milhões de informações de outros usuários também conectados. A seguir, o cliente envia sua lista de *downloads*, com os arquivos que ele deseja baixar. O servidor retorna uma lista de outros clientes que possuem tais arquivos – ou ao menos parte deles. O cliente também enviará frequentemente ao servidor buscas por determinados arquivos. O servidor, então, retornará os resultados encontrados, o cliente selecionará os arquivos que deseja, e pedirá ao servidor a lista de fontes – outros clientes que possuem o arquivo desejado. O servidor retorna ao cliente os endereços de IP e as portas de comunicação TCP (a porta padrão é a 4662) de cada fonte de forma que o usuário possa realizar uma conexão com outro cliente para realizar o *download*.

O servidor também pode rejeitar um cliente pelo fato dele ter uma ID baixa ou por ter esgotado sua capacidade física.

O cliente se conectará a outros servidores somente para obter mais fontes para um dado arquivo que deseja baixar. Tais conexões serão feitas através de protocolo UDP e não TCP.

O UDP também será utilizado entre cliente e servidor para verificar periodicamente o status e descrição dos servidores da sua lista de servidores, mantendo-a atualizada. Essa atualização (Keep-Alive) irá gerar cerca de uma dúzia de pacotes por hora. Quando o cliente requisitar o status do servidor a resposta será o número de usuários a ele conectados e o número de arquivos nele listados. Em seguida o usuário envia um pedido de descrição ao servidor que responderá o seu nome e uma pequena string de descrição.

Com as informações vindas do servidor o cliente irá realizar uma conexão TCP com outros clientes para realizar o *download* de um arquivo. Um mesmo arquivo poderá ser baixado de várias fontes simultaneamente, pois cada arquivo é dividido em partes e cada uma dessas partes é fragmentada de forma que se possa obter um fragmento de cada fonte. Dessa aumenta-se a velocidade com que um arquivo é baixado e também sua disponibilidade na rede para *download*, pois poder-se baixar o fragmento de uma fonte mesmo que ela não tenha o arquivo completo, basta ter um fragmento completo de um arquivo para que esse segmento possa ser compartilhado para todos os demais usuários.

Ao realizar a conexão com uma fonte para baixar determinado arquivo, o cliente deverá entrar numa fila. Se a fila da fonte estiver vazia, o *download* poderá começar de imediato. Do contrário o cliente terá que esperar até que os outros usuários na sua frente na fila realizem o *download* da parte desejada. Quando o cliente atinge o início da fila, a fonte inicia uma conexão para enviar o arquivo para ele. Caso o cliente já tenha obtido o fragmento que a fonte está para enviar, ele apenas recusa o *upload* pois quando o eMule obtém determinada parte do arquivo, ele não notifica às fontes existentes.

Se um cliente possuir uma ID baixa significa que ele não possui um endereço IP público. Dessa forma, outros clientes (fontes) não conseguem se conectar a ele diretamente e toda a comunicação deverá ser feita através do servidor. Isso aumenta o gasto computacional no servidor o que acaba gerando uma relutância destes em aceitar usuários com ID baixa. Além disso, um cliente com ID baixa não poderá se conectar a outro cliente também com ID baixa de outro servidor, pois o eMule não permite esse tipo de tunelamento de requisição de arquivo entre servidores.

Para dar suporte a clientes com ID baixa, o eMule utiliza um sistema de *callback* (rechamada) onde uma fonte com ID alta pode pedir através do servidor

que o cliente com ID baixa se conecte a ele para que possam trocar arquivos. Se um cliente A, que possui uma ID alta, deseja baixar um arquivo do cliente B, que possui uma ID baixa, estando ambos conectados a um mesmo servidor, A envia ao servidor um pedido de *callback* solicitando que B se conecte a ele. Em seguida, o servidor envia a B, através da conexão TCP já existente entre eles, o pedido de *callback* enviando o endereço IP e a porta de A de forma que B possa se conectar ao cliente A e enviar o arquivo sem sobrecarregar o servidor. Apenas um cliente com ID alta pode fazer um pedido de *callback* ao servidor para um cliente de ID baixa. Um cliente de ID baixa não pode fazer o mesmo pedido para se conectar a outro cliente de ID baixa, pois não haverá como conectar ambos os pontos diretamente sem a intermediação do servidor.

O eMule possui um sistema de créditos para incentivar o compartilhamento de arquivos. Quanto mais arquivos um cliente envia para outros clientes, mais créditos ele ganha e mais rápido ele irá avançar nas suas filas de espera. Para implementar esse sistema de créditos adotou-se o conceito de identificador do usuário (*User ID*), que é um índex global e único (GUID - Globally Unique Identifier) de 128 bits. Esse *User ID* identifica o usuário mesmo através de seções uma vez que identifica o computador.

Para não permitir que alguma pessoa se passe por outra, o eMule assegura o sistema de créditos usando criptografia de chave pública RSA. Este sistema de créditos, assim como a troca de informações gerais (atualizações de listas e fontes) e melhoria de desempenho são realizados utilizando uma extensão do protocolo do eDonkey.

O cliente usará protocolo UDP para se comunicar com outros clientes essencialmente para obter o seu posicionamento nas filas. Ao enviar o pedido de posicionamento na fila, o cliente poderá receber três tipos de respostas vindas da fonte: *ack* contendo a posição dele na fila de espera da fonte, uma mensagem de que a fila está cheia, ou uma mensagem que a fonte não possui o item especificado em sua lista de arquivos.

O eMule identifica um arquivo não pelo seu nome, obviamente, mas por um identificador de arquivo ou *File ID*. Esse ID serve tanto para identificar um arquivo de forma única na rede como também para detectar um eventual erro, sendo normalmente, uma GUID computada através de um *hash* do conteúdo de arquivo.

Existem dois tipos de ID de arquivo: o *hash* do arquivo (*file hash*) e o *hash* de raiz do arquivo (*root hash*).

O *File Hash* é um GUID (*Globally Unique Identifier*) de 128 bits calculado através de um *hash* do conteúdo do arquivo. O eMule divide o arquivo em partes de 9,28MB e calcula o GUID de cada parte aplicando o algoritmo MD4. Em seguida os *hashes* são combinados em um único GUID que será o identificador do arquivo. O link do arquivo dentro da rede eD2k (eDonkey2000) contém o identificador do arquivo como é mostrado a seguir:

- ed2k://|file|name|12043984|6744FC42EDA527B27F0B2F2538728B3E|/
- o arquivo descrito no link acima possui 12043984 bytes e seu identificador é 6744FC42EDA527B27F0B2F2538728B3E.

Para garantir que o *hash* será enviado corretamente, o link para o arquivo poderá conter o hash de cada uma das partes do arquivo, além do GUID o arquivo como mostra o exemplo abaixo:

- ed2k://|file|name|12043984|6744FC42EDA527B27F0B2F2538728B3E|
  p=264E6F6B587985D87EB0157A2A7BAF40:17B9A4D1DCE0E4C2B672DF257145E98A|/
- o arquivo acima possui 12043984 bytes, logo possui duas partes e, consequentemente, o que se denomina *hashset* contém dois *hashes*. O identificador do arquivo continua sendo '6744FC42EDA527B27F0B2F2538728B3E' e 'p' é o *hashset*, onde o *hash* de cada parte é mostrado separado por ':'. Logo, temos que o *hash* da primeira parte do arquivo é '264E6F6B587985D87EB0157A2A7BAF40' e o da segunda parte é '17B9A4D1DCE0E4C2B672DF257145E98A'.

Os exemplos acima foram retirados do site http://www.emule-project.net/home/perl/help.cgi?l=1&rm=show_topic&topic_id=589.

Ao fazer o *download* do arquivo, conforme cada parte é completa, calcula-se o *hash* dela e compara-se com o vindo do cliente fonte. Se o *hash* recebido e o *hash* calculado forem idênticos, a parte recebida do arquivo está íntegra. Caso contrário, a parte recebida está corrompida e precisa ser reenviada. Para não ter que reenviar os 9,28MB, o eMule faz com que sejam enviados blocos de 180 KB e então seja recalculado o *hash* no destino até que os dois *hashes* se igualem. No melhor caso,

apenas 180 KB serão reenviados, e 9,28 MB no pior caso. Esse sistema é chamado de ***ICH - Intelligent Corruption Handling.***

Outra forma de aumentar a segurança é o sistema de *Root Hash* que incorpora o sistema Avançado ICH (***AICH - Advanced Intelligent Corruption Handling***) do eMule. O ICH só é capaz de validar a parte inteira dos 9,28MB da parte do arquivo e nenhum bloco menor. Clientes maliciosos podem difundir dados corrompidos na rede ou blocos inteiros falsos de arquivos que não seriam identificados pelo IHC. No AICH, cada parte do arquivo é dividida em blocos de 180 KB, gerando 53 blocos por parte. Para cada um desses blocos gera-se um *hash* através do algoritmo SHA1. Esses valores formarão o nível mais baixo da árvore de *hashes* ou AICH *Hashset*. A partir dos *hashes* de cada bloco de 180 KB, pode-se criar um identificador para a parte do arquivo de 9,28MB que ficará alguns níveis acima no AICH *Hashset* e, partindo do *hash* de cada parte do arquivo, pode-se criar também o *Root Hash* do arquivo como raiz do AICH *Hashset*. O link de um arquivo pode conter o *Root Hash* como mostrado abaixo:

- ed2k://|file|name|12043984|6744FC42EDA527B27F0B2F2538728B3E|
  h=A2NWOTYURUU3P3GCUB6KCNW3FTYYELQB|/
- o arquivo acima possui 12043984 bytes, seu ID é 6744FC42EDA527B27F0B2F2538728B3E e 'h' é o seu *Root Hash*. Arquivos que contenham o *Root Hash* no link do arquivo são mais confiáveis. Caso não haja, o Emule obterá o *Root Hash* de outros clientes, só aceitando como verdadeiro se obtiver o mesmo *Root Hash* vindo de 10 pontos diferentes.

Quando o eMule detecta uma parte de um arquivo corrompida, pede o Pacote de Recuperação da parte corrompida para algum cliente aleatório que possua o AICH *hashset* completo. Esse pacote contém os *hashes* dos 53 blocos da parte corrompida e um número 'x' de *hashes* de verificação tal que 2^X é maior ou igual ao número de contagem da parte corrompida. A ilustração que segue ajuda a compreender o funcionamento do Pacote de Recuperação da parte do arquivo.

Na figura 3.7 o nó 2 pode ser obtido a partir dos 53 Hashes de Bloco (folhas verdes da árvore).Para obter o Nó amarelo do nível 1 da árvore, entretanto será preciso o hash da parte 1 (que está em verde), e para obter o root hash será preciso o nó verde (na direita) no nível 1 da árvore.

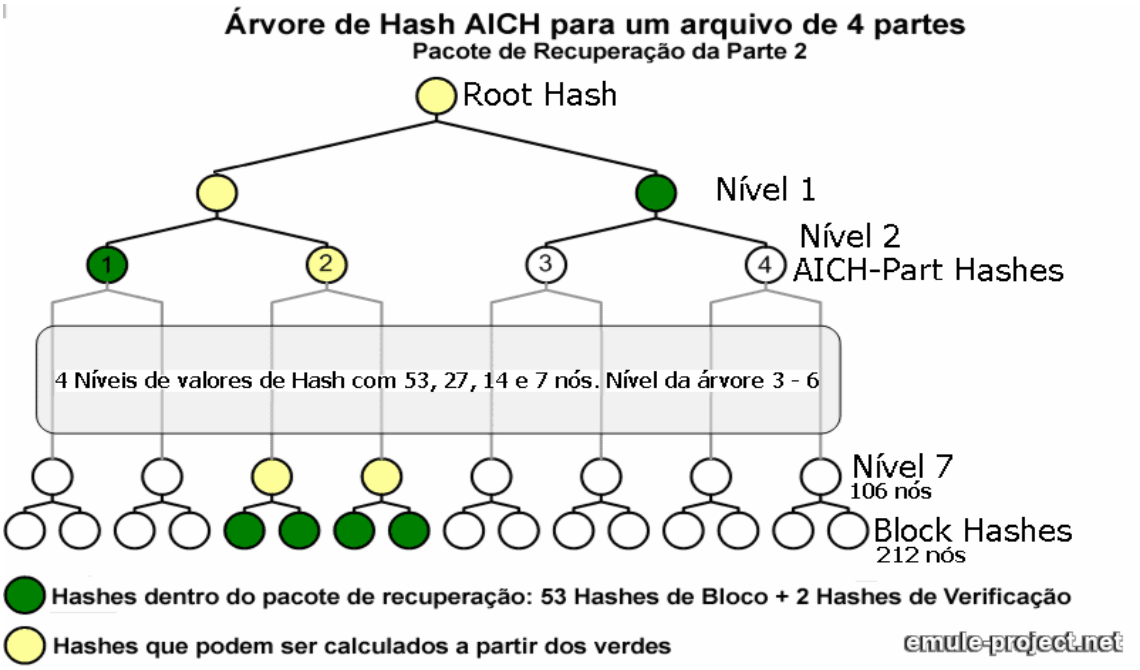


Figura 3.7: Árvore de Hash AICH

Uma busca por arquivo é sempre iniciada pelo usuário e é razoavelmente simples. O cliente requisita a busca e recebe uma resposta do servidor com o resultado da busca. Caso o resultado seja muito grande, ele virá comprimido. Em seguida, o cliente escolhe os arquivos que deseja fazer o *donwload* e pede ao servidor a lista de fontes. O servidor pode enviar uma mensagem de status antes de enviar a lista de fontes ao cliente, informando o número de usuários conectados àquele servidor e os tipos de arquivos suportados por ele. Depois de verificar que as fontes são novas, o cliente inicia uma tentativa de conexão e as adiciona em sua lista de fontes.

O eMule usa conexões UDP para melhorar os resultados da busca. O cliente, ao realizar a busca, além de requisitar a busca no servidor em que está conectado via TCP, pode enviar pedidos de busca aos outros servidores de sua lista via UDP e obter mais resultados para a busca. Após receber as respostas e selecionar os arquivos para realizar o *download*, o cliente também buscará nos outros servidores via UDP novas fontes para o arquivo desejado, aumentando as possibilidades de recepção de partes do arquivo.

4 COMPARAÇÃO DE REDES P2P

O Napster é considerado P2P porque os endereços de nós Napster transpassam o sistema DNS no sentido em que o servidor Napster resolve os endereços IP dos PC que tenham um determinado arquivo MP3, transferindo o controle das transferências de arquivo para os nós. Dos modelos vistos, é o único que utiliza servidor centralizado.

Diferentemente do Napster, o Gnutella surgiu como solução para compartilhamento de qualquer tipo de arquivo e não só MP3. Uma das desvantagens desse modelo está no fato de que cada nó deve armazenar algumas informações sobre quais pacotes foram vistos em certo período de tempo. Para poder manter um bom histórico do roteamento de pacotes, é necessário que o nó possua uma boa disponibilidade de memória. Por outro lado, se o nó possuir pouca memória pode ter que interromper encaminhamentos de pacotes que foram previamente enviados, criando tráfego desnecessário na rede.

Aplicativos baseados em FreeNet são os de pior interface com o usuário. Apesar disso, tecnicamente, é o modelo de redes peer-to-peer mais promissor pois, nesse tipo de rede, é dada ênfase à questão do roteamento de requisições pelo uso de uma política de replicação dos dados para que estes fiquem sempre mais próximos do usuário final, o que o torna o modelo de maior escalabilidade dentre os estudados. Na figura 4.1 pode-se observar que com uso dessa técnica, ao longo do tempo o desempenho de uma rede FreeNet tende a ser melhor que as demais. A preocupação com esse modelo é que devido às suas características, existe grande possibilidade de mau uso da rede, pois o FreeNet é fortemente baseado em criptografia e anonimidade. Nesse modelo, toda transmissão de qualquer tipo de arquivo é sigilosa tanto no sentido do transmissor quanto do receptor, e, ainda, quanto ao conteúdo armazenado, o que o torna convidativo para atividades ilegais de pirataria, por exemplo.

Da figura 4.2 tem-se que baseado em dados de 2001, a percentagem de usuários Napster conectados por modem (64kbps ou menos) era de aproximadamente 25% da rede enquanto que esse mesmo tipo de usuário na rede Gnutella representa apenas 8%. Na figura 4.3 pode-se observar que os nós da rede Napster são ligeiramente mais consistentes e oferecem menor variação no número de arquivos compartilhados que os nós da rede Gnutella e que aproximadamente 40

a 60% dos nós compartilhavam somente entre 5 a 20% dos arquivos disponibilizados.

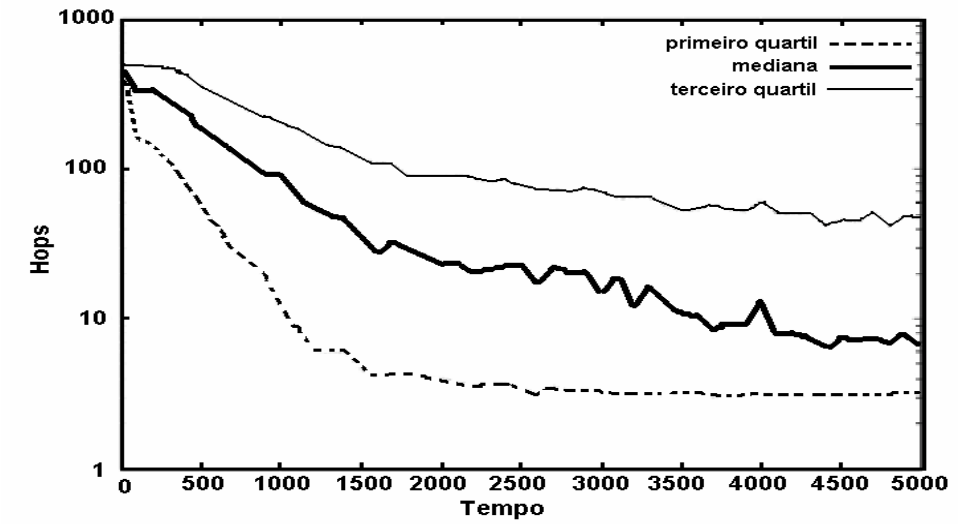


Figura 4.1: Evolução da rede FreeNet em função do tempo e do número de caminhos percorridos (hops) Fonte: CLARK, I. et al, p.13.

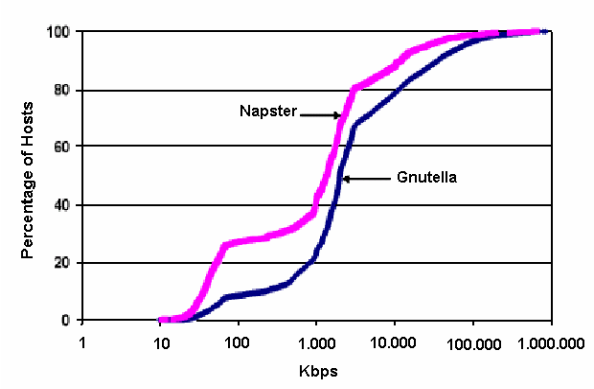


Figura 4.2: Gargalo de mensagens em relação ao número de hosts da rede e a velocidade de conexão do usuário.
Fonte: SAROIU, S.; GUMMADI, P. K.; GRIBBLE, S. D.,2001. p.7.

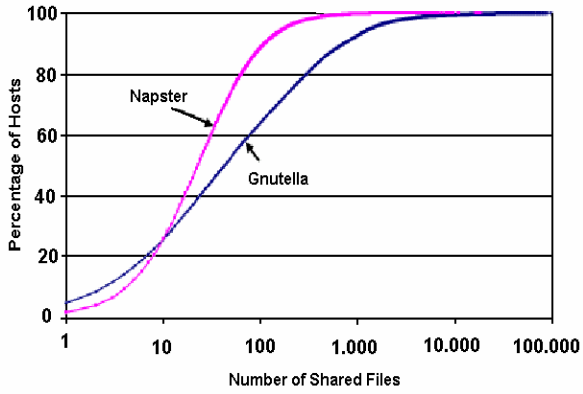


Figura 4.3: Número de arquivos compartilhados em relação ao número de hosts da rede.
Fonte: SAROIU, S.; GUMMADI, P. K.; GRIBBLE, S. D.,2001. p.9.

O Emule concatenou as qualidades das três (Napster, Gnutela e FreeNet) numa necessidade de melhorar os aplicativos, aumentando sua eficiência e tornando-a mais amigável para o usuário P2P. Realiza inicialmente um serviço de indexação centralizado diminuindo o problema de desempenho de comunicação de

rede (Napster); fornece um bom nível de anonimanidade (Gnutela); utiliza função Hash Sha1; e usa a política de replicação dos dados melhorando a escalabilidade (FreeNet). Esse modelo também tem preocupação com o mau uso da rede, que é baseado em criptografia e o transmissor e o receptor são sigilosos. Seu maior problema é o sistema de créditos de compartilhamento.

Analisando algumas das características das redes peer-to-peer nos modelos escolhidos podemos afirmar:

• **Anonimidade**: No Napster é quase inexistente, só existe se considerarmos a anonimidade entre os nós. Sistemas baseados em Gnutella só possuem anonimidade no que se refere à publicação enquanto que em sistemas baseados em FreeNet e Emule a anonimidade é garantida quanto à publicação, ao acesso, à disponibilização e ao documento.

• **Escalabilidade**: No Napster é quase inexistente pois todos os nós têm que estar conectados a um servidor central ao passo que no FreeNet e no Gnutella ela pode, em teoria, se expandir indefinidamente na proporção do aumento da rede. No Emule Utiliza função Hash e usa a política de replicação dos dados melhorando a escalabilidade. Na prática, um grande volume de mensagens pode tirar essas redes de serviço.

• **Segurança:** No Napster é dependente da instaurada no servidor. No Gnutella é baixa pois seu protocolo foi desenhado para se voltar mais para a funcionalidade e eficiência. No FreeNet e no Emule a segurança quanto ao conteúdo é alta se utilizada chaves do tipo Content-Hash Keys.

• **Auto-organização**: Característica dispensável na arquitetura Napster pois toda organização da rede é feita pelos servidores centrais. Já os modelos FreeNet, Gnutella e Emule conseguem se adaptar à natureza dinâmica da Internet pelo uso de mensagens periódicas de Ping para detecção dos nós disponíveis a cada momento. Isso traz um custo muito alto no que diz respeito à quantidade de mensagens trafegadas nessas redes.

• **Resistência à falhas**: No Napster é dependente do balanceamento dos servidores da rede. Se a falha ocorrer em servidor central de alta requisição, toda a rede pode ficar comprometida. No Gnutella, no FreeNet e no Emule o mau funcionamento de alguns nós não acarreta necessariamente na queda da rede se ainda houver quantidade suficiente de nós conectados no dado momento, mas degrada a performance.

5 CONCLUSÃO

A tecnologia peer-to-peer (P2P) surgiu para mudar o paradigma existente em compartilhamento de informação, à medida que não depende de uma organização central ou hierárquica, além de dispor aos seus integrantes as mesmas capacidades e responsabilidades.

O Napster, fortemente apoiado no uso de servidores, apesar de útil e funcional, exige um controle muito grande e eficiente com relação ao conteúdo da rede, pois os sistemas baseados nesse modelo são bastante suscetíveis à troca de arquivos protegidos por direitos autorais.

O modelo FreeNet, escorado pela bandeira da liberdade de expressão, trouxe grande contribuição para a solução de sistemas fortemente baseados em anonimidade mas, por outro lado, essa mesma característica pode tornar falha a segurança dos sistemas pelo total desconhecimento que se tem dos outros nós que participam da rede e, dependendo da chave que se utiliza para recuperação de um arquivo, da impossibilidade de discernir a qualidade do que se está recuperando da rede. O compartilhamento seguro nesse tipo de rede torna indispensável o prévio conhecimento do código (chave) do arquivo que se deseja, o que restringe e retarda a dinâmica do processo.

O modelo Emule também é baseado em anonimidade, utiliza sistema de créditos para incentivar o compartilhamento de arquivos e tem um tratamento de dados muito interessante que é a árvore de rash. Suas maiores vantagens são a agilidade na busca de dados e o sistema avançado de tratamento inteligente de corrompimento (AICH).

Todos os modelos vistos possuem deficiências em uma ou outra característica e adaptações na arquitetura são implementadas para tentar contornar esses problemas. Um exemplo claro disso é o modelo Gnutella que devido ao seu mecanismo de busca ser por inundação, confere fraco desempenho ao sistema, mas, evoluiu para uma variação do modelo denominada FastTrack [FASTRACK 2005] [ARAÚJO 2003], que, distribuindo quantitativamente os pontos da rede servidos com supernós, consegue solucionar essa falha, pois, sendo as pesquisas realizadas no nível do supernó, as mensagens não necessitam se propagar por todos os nós da rede.

6 REFERÊNCIAS

ANDROUTSELLIS-THEOTOKIS, S.; SPINELLIS, D. **A Survey of Peer-to-Peer content distribution technologies**. Athens University of Economics and Business, Athens, v.36, n.4, p. 335-371, Dec. 2004.

ARAÚJO, G. **Tecnologias P2P e seu impacto na rede**. 2003. 19 f. Trabalho apresentado no 9º Seminário de Capacitação Interna da RNP - Universidade Federal da Bahia, Salvador.

BERKES, J. E. **Decentralized Peer-to-Peer Network Architecture**: Gnutella and Freenet. Winnipeg: University of Manitoba, 2003. p.3-5. Disponível em: <www.sysdesign.ca/archive/berkes_gnutella_freenet.pdf>. Acesso em: jun. 2007.

BIGUS, J. P. **Constructing Intelligent Agents Using Java**: New York: John Wiley & Sons, Inc, 2001.

CAMARGO, E. **Um Estudo sobre Sistemas P2P**. 2004. Disponível em: < www.ime.usp.br/~yw/2004/mac5701i/monografias/emilio-reverbel.pdf>. Acesso em jul. 2007.

CLARK, I. et al. **FreeNet: A distributed anonymous information storage and retrieval system**. Disponível em: <//freenet.sourceforge.net/freenet.pdf>. Acesso em: jun. 2007.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Distributed system: concepts and design**. 3rd. ed. Boston: Addison-Wesley, 2000.

DAHLIA, M. and David Ratajczak.**Viceroy: A scalable and dynamic emulation of the butterfly**. Proc. SIGCOMM, ACM press pages 183-192, 2002.

DIEGO, D. and Donal O´ Mahony. **Overlay networks – a scalable alternative for p2p. IEEE internet computing**, pages 79-82, August 2003

DING, C. H.; NUTANONG S.; BUYYA R. **P2P Networks for content sharing**. 2003. 35 f. Technical Report – Department of Computer Science and Software Engineering, The University of Melbourne – Australia.

DURANTE, G. B. **Redes Peer-to-Peer**. 2004. Disponível em: <www.gta.ufrj.br/grad/04_1/p2p/index.html>. Acesso em jul. 2007.

EMULE, **Project.net** disponível em: <www.emule-project.net/home/perl/general.cgi?l=30&rm> Acesso em: jul 2007.

FASTRACK.2005.Disponível em <www.fasttrackdcn.net> Acesso em: jul. 2007.

FREENET. Disponível em: <www.freenetproject.org> Acesso em: jul. 2007.

GNUTELLA. Disponível em: <www.gnutella.com>. Acesso em: ago. 2007.

GRANVILLE, L. Z. G. et al. **Tecnologia Peer-To-Peer Aplicada no Gerenciamento de Redes Ópticas**. 2005. 06 f. – Instituto de Informática, UFRGS, Porto Alegre.

HEYLIGHEN, F. **Self Organization**. 1997. Disponível em: <//pespmc1.vub.ac.be/SELFORG.html>. Acesso em: ago. 2007.

iMesh. Disponível em: <www.imesh.com>. Acesso em: ago. 2007.

JXTA. Disponível em: <www.jxta.org>. Acesso em: ago. 2007.

MILOJICIC, D. C. **Peer-to-Peer Computing.** Palo Alto. HP Laboratories, 2002

NASPTER.2005. Disponível em: <www.napster.com>. Acesso em: abr. 2007.

ORAM, A. **Peer-to-Peer o poder transformador das redes ponto a ponto**. 2ª. ed. São Paulo: Berkeley, 2001.

PETAR M. and David M. **Kademlia: A peer-topeer information system based on the xor metric.** Proceedings of the Twenty-first ACM symposium on Principles of Distributed Computing (PODC 2002), pages 108-117, july 2002.

RATNASAMY, S. Hellerstein, : **Range queries over DHTs**. 2003. Technical Report IRB-TR-03-009, Intel.

RIBEIRO, P. R. **Aplicações educacionais colaborativas em redes P2P: Avaliação de mecanismos para registros de anotações em grupos**. 2003. 13 f. Projeto de Pesquisa - Instituto de Engenharia Elétrica, UNICAMP, São Paulo.

RIPEANU, M. **Peer-to-Peer Architecture Case Study: Gnutella Network**. Chicago: The University of Chicago, 2002. p.5-8. Disponível em: <//people.cs.uchicago.edu/~matei/PAPERS/gnutella-rc.pdf>. Acesso em: jul. 2007.

ROCHA, J. et al. **Peer-to-Peer: Computação colaborativa na Internet. In:** SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, 22., 2004, Gramado. II/UFRGS, 2004. 45 f.

ROWSTRON A. **Pastry :Scalable, distributed object location and routing for largescale pee-to-peer systems .**Middleware 2001, pages 108-127.

SAROIU, S.; GUMMADI, P. K. GRIBBLE, S. D. **A Measurement Study of Peer-to-Peer File Sharing Systems**. 2001. 15 f. Technical Report – Department of Computer Science & Engineering, University of Washington, Seatle – USA.

SETI INSTITUTE. 2005. Disponível em: <//setiathome.ssl.berkeley.edu >. Acesso em: jun. 2007.

SILVA, W. R. S.: **Introdução às redes peer-to-peer**. 2003. Disponível em: <//www.gta.ufrj.br/seminarios/semin2003_1/william/index.htm>. Acesso em: jul. 2007.

STOICA, I., MORRIS R., KARGER, D. KAASHOEH, M., and BALAKRISHNAN, h. : **Chord : A scalable peer-to-peer lookup service for internet application,** Proceeding of the 2001conference on application technologies, architectures, and protocols for computer communications, páginas 1-12.

WIKIPEDIA. **Peer-to-Peer.** 2005.Disponível em: <//en.wikipedia.org/wiki/Peer-to-peer >. Acesso em: jun. 2007.

WIKIPEDIA. **Emule.** 2005.Disponível em: < //pt.wikipedia.org/wiki/EMule>. Acesso em: jun. 2007

WIKIPEDIA. **File Sharing.** Disponível em: <//en.wikipedia.org/wiki/File_sharing#Generational_classification_of_peer-to-peer_file_sharing_networks >. Acesso em: abr. 2007.